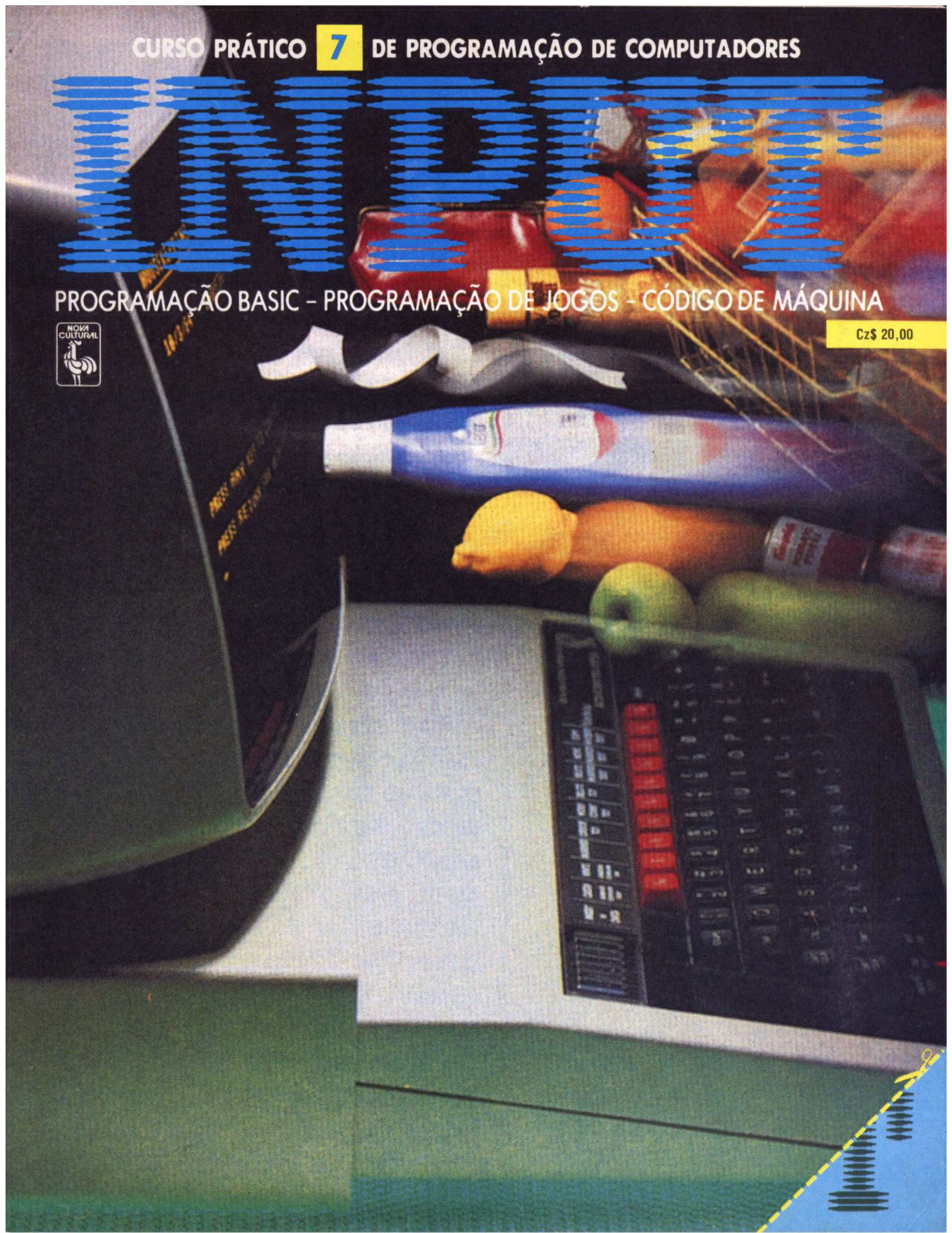
CURSO PRÁTICO 7 DE PROGRAMAÇÃO DE COMPUTADORES

CP/M

PROGRAMAÇÃO BASIC - PROGRAMAÇÃO DE JOGOS - CÓDIGO DE MÁQUINA

NOVA CULTURAL

Cz$ 20,00



INPUT

Vol. 1 N.º 7

NESTE NÚMERO

PROGRAMAÇÃO DE JOGOS

BOMBARDEIOS E EXPLOSÕES

Explosões e incêndios são parte integrante de muitos videogames de ação. Aprenda a reproduzir esses fenômenos no vídeo do computador. Um programa simples de bombardeios aéreos 121

PROGRAMAÇÃO BASIC

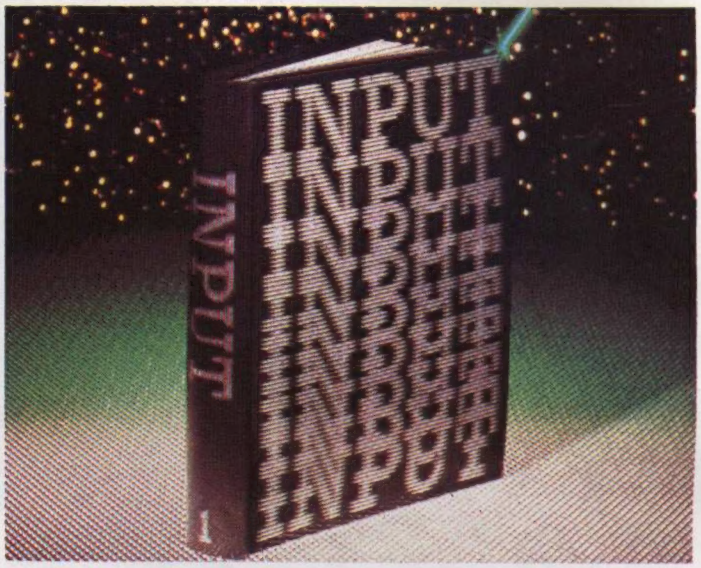
OS COMANDOS READ E DATA

Como fazer com que o micro leia dados armazenados internamente em declarações **DATA**, evitando a digitação de longos e exaustivos programas. Veja como utilizar essa técnica para a realização de gráficos, listas, índices e muitas outras coisas 128

APLICAÇÕES

PONHA ORDEM EM SUAS CONTAS

Use seu micro para fazer cálculos e manter registros financeiros. Nisso, ele é tão bom quanto os grandes computadores comerciais. Atualize os registros, verifique o balanço e imprima os resultados 134



PLANO DA OBRA

"INPUT" é uma obra editada em fascículos semanais, e cada conjunto de 15 fascículos compõe um volume. A capa para encadernação de cada volume estará à venda oportunamente.

COMPLETE SUA COLEÇÃO

Exemplares atrasados, até seis meses após o encerramento da coleção, poderão ser comprados, a preços atualizados, da seguinte forma: **1. Pessoalmente** — por meio de seu jornaleiro ou dirigindo-se ao distribuidor local, cujo endereço poderá ser facilmente conseguido junto a qualquer jornaleiro de sua cidade. Em São Paulo os endereços são: Rua Brigadeiro Tobias, 773 (Centro); Av. Industrial, 117 (Santo André); e, no Rio de Janeiro: Rua da Passagem, 93 (Botafogo). **2. Por carta** — Poderão ser solicitados exemplares atrasados também por carta, que deve ser enviada para DINAP — Distribuidor Nacional de Publicações — Números Atrasados — Estrada Velha de Osasco, 132 (Jardim Tereza) — CEP 06000 — Osasco — São Paulo. **3. Por telex** — Utilize o nº (011) 33670 ABSA. Em **Portugal**, os pedidos devem ser feitos à Distribuidora Jardim de Publicações Ltd. — Qta. Pau Varais, Azinhaga de Fetais — 2685, Camarate — Lisboa; Tel. 257-2542 — Apartado 57 — Telex 43 069 JARLIS P.
Não envie pagamento antecipado. O atendimento será feito pelo reembolso postal e o pagamento, incluindo as despesas postais, deverá ser efetuado ao se retirar a encomenda na Agência do Correio. **Atenção**: Após seis meses do encerramento da coleção, os pedidos serão atendidos, dependendo da disponibilidade de estoque. **Obs.:** Quando pedir livros, mencione sempre o título e/ou o autor da obra, além do número da edição.

COLABORE CONOSCO

Encaminhe seus comentários, críticas, sugestões ou reclamações ao Serviço de Atendimento ao Leitor — Caixa Postal 9442, São Paulo — SP.

NOVA CULTURAL

Editor
VICTOR CIVITA

REDAÇÃO

Diretora Editorial: Iara Rodrigues

Editor chefe: Paulo de Almeida
Editor de texto: Cláudio A.V. Cavalcanti
Editor de Arte: Eduardo Barreto
Chefe de Arte: Carlos Luiz Batista
Assistentes de Arte: Ailton Oliveira Lopes, Dilvacy M. Santos, José Maria de Oliveira, Grace A. Arruda, Monica Lenardon Corradi
Secretária de Redação/Coordenadora: Stefania Crema
Secretários de Redação: Beatriz Hagström, José Benedito de Oliveira Damião, Maria de Lourdes Carvalho, Marisa Soares de Andrade, Mauro de Queiroz
Secretário Gráfico: Antonio José Filho

COLABORADORES
Consultor Editorial Responsável: Dr. Renato M.E. Sabbatini (Diretor do Núcleo de Informática Biomédica da Universidade Estadual de Campinas)
Execução Editorial: DATAQUEST Assessoria em Informática Ltda. Campinas, SP.
Tradução: Maria Fernanda Sabbatini

Adaptação, programação e redação: Aluísio J. Dornellas de Barros, Marcelo R. Pires Therezo, Raul Neder Porrelli
Coordenação geral: Rejane Felizatti Sabbatini
Assistente de Arte: Dagmar Bastos Sampaio

COMERCIAL
Diretor Comercial: Roberto Martins Silveira
Gerente Comercial: Flávio Ferrucio Maculan
Gerente de Circulação: Denise Maria Mozol

PRODUÇÃO
Gerente de Produção: João Stungis
Coordenador de Impressão: Atílio Roberto Bonon
Preparador de Texto/Coordenador: Eliel Silveira Cunha

Preparadores de Texto: Ana Maria Dilguerian, Antonio Francelino de Oliveira, Karina Ap. V. Grechi, Levon Yacubian, Maria Teresa Galluzzi, Paulo Felipe Mendrone
Revisor/Coordenador: José Maria de Assis
Revisoras: Conceição Aparecida Gabriel, Isabel Leite de Camargo, Ligia Aparecida Ricetto, Maria do Carmo Leme Monteiro, Maria Luiza Simões, Maria Teresa Martins Lopes.

© Marshall Cavendish Limited, 1984/85.
© Editora Nova Cultural Ltda., São Paulo, Brasil, 1986.
Edição organizada pela Editora Nova Cultural Ltda.
(Artigo 15 da Lei 5 988, de 14/12/1973).

Esta obra foi composta na AM Produções Gráficas Ltda. e impressa na Divisão Gráfica da Editora Abril S.A.

BOMBARDEIOS E EXPLOSÕES

- COMO CRIAR FLASHES NA TELA
- UM PROGRAMA SIMPLES DE BOMBARDEAMENTO AÉREO
- ENRIQUEÇA A AÇÃO COM CHAMAS E EXPLOSÕES

Faça de seus jogos de ação um eletrizante passatempo, aprendendo a obter efeitos gráficos de explosões e incêndios na tela do micro.

É relativamente fácil tornar seus programas de jogos mais espetaculares e emocionantes: para isso, basta acrescentar algumas rotinas com efeitos gráficos especiais. E elas não precisam ser muito complexas para dar a esses jogos um dinamismo bem diferente.

Como você verá nesta lição, existem muitas maneiras de produzir efeitos visuais em BASIC. É preciso apenas saber usar o tipo certo de efeito para o jogo que está sendo programado.

Os efeitos gráficos para chamas e explosões explicados aqui são adequados a todos os tipos de jogos que envolvem a destruição de objetos como construções, carros, aviões, tanques, navios, etc. Alguns deles ficarão ótimos também em jogos espaciais.

Entretanto, como os efeitos de chamas e explosões utilizam o conceito de blocos gráficos, o tamanho máximo do alvo sobre o qual serão colocados é restringido pelas dimensões desses efeitos. Isso significa que você não pode adicioná-los a um programa qualquer; pelo contrário, essa inclusão deve ser cuidadosamente planejada.

Por outro lado, muitos computadores têm comandos embutidos no BASIC que permitem a obtenção de flashes (inversão rápida de cores no vídeo), que são suficientes para muitos tipos de jogos, como os de guerra espacial, e que têm a vantagem de não exigir adaptações para uso em um jogo determinado.

S

Vamos mostrar aqui como criar, nos micros compatíveis com o Sinclair Spectrum (como o TK-90X), o efeito visual

de um incêndio provocado pela explosão de uma bomba; depois de ir romper fortemente, o fogo se extingue gradualmente, desaparecendo da tela à medida que o prédio desaba. Isso será explicado mais adiante.

Mas, primeiro, precisamos de um avião e de uma bomba. Portanto, digite o programa abaixo:

```
10 FOR N=USR "p" TO USR "q"+7
20 READ a
30 POKE n,a
40 NEXT n
50 DATA 32,16,136,154,155,8,
16,32
60 DATA 0,16,16,120,28,28,0,0
```

Usamos aqui a técnica convencional de blocos gráficos definidos pelo usuário (UDG), disponível no Spectrum. Assim, criamos imagens tanto para o avião quanto para a bomba, armazenadas em uma área especial, definida pelos endereços devolvidos pelas expressões **USR "p"** e **USR "q"** na linha 10. Os códigos correspondentes estão nas declarações **DATA** das linhas 50 e 60, e são colocadas na memória com o **POKE** da linha 30. Ao rodar o programa, nada acontecerá de início. Para verificar se os códigos gráficos em **DATA** foram digitados corretamente, digite a linha de comando abaixo (sem precedê-la com um número de linha).

```
PRINT AT 10,15;CHR$ 159;" ";
CHR$ 160
```

— que mostrará na tela os dois blocos gráficos definidos.

Em seguida, você precisará de um prédio ou casa para bombardear. Não há necessidade de um novo programa para isto. Apenas modifique as linhas 10 e 50 do programa anterior.

```
10 FOR N=USR "r" TO USR "r"+7
20 READ a
30 POKE n,a
40 NEXT n
50 DATA 255,153,255,153,255,
153,255,255
```

Execute o programa de novo e teste-o, digitando esta linha (novamente, sem colocar número de linha antes):

```
PRINT AT 20,15; CHR$ 161
```

Tendo ficado satisfeito com o resultado visível na tela, digite **NEW** para apagar o programa da memória. Os blocos gráficos definidos antes ficarão armazenados na memória RAM do computador, a menos que você o desligue.

COMECE O BOMBARDEIO

Entre agora o programa do bombardeio por meio das linhas seguintes:

```
10 BORDER 0: PAPER 5: INK 0:
 CLS
 20 LET a$="                "
200 PRINT  PAPER 4;AT 20,0;a$;
a$;a$;  a$
210 PRINT  INK 1;AT 19,12;CHR$
161;   CHR$ 161;CHR$ 161
```

O que estas linhas fazem, como você verá ao executá-las com o comando **RUN**, é apenas desenhar uma faixa verde, com dezesseis pixels de altura, na parte de baixo da tela; sobre ela erguese um armazém ou coisa parecida. A forma como isso é feito ilustra bem o uso de variáveis alfanuméricas no trabalho com gráficos.

Como você precisa de 64 bloquinhos coloridos de verde para fazer o terreno gramado, a linha 20 coloca em a$ uma seqüência de dezesseis espaços em branco. Em seguida, a linha 200 a imprime na tela quatro vezes, começando na linha 20 e prosseguindo na linha 21. Isso economiza a digitação de 64 a$!

A construção é colocada na tela por meio de um quatro **PRINT** do caractere gráfico de código 161 (que é o associado à tecla r, no computador, conforme a definição no programa anterior).

Para fazer o avião voar, só é preciso incluir no programa as seguintes linhas:

```
215 PAUSE 100
220 LET ay=6: LET by=ay
230 FOR x=0 TO 30
240 PRINT AT ay,x;" ";CHR$ 159
250 LET bx=x
260 IF by<19 THEN  PRINT AT by
+1,bx+1;CHR$ 160;AT by,bx;" "
270 LET by=by+1: LET bx=bx+1
280 IF x>29 THEN  PRINT AT ay,
x+1;" "
290 NEXT x
```

Não existe nada de incomum nessa seção do programa: apenas as técnicas convencionais de movimentação de um bloco gráfico na tela, que temos usado em praticamente todos os programas de jogos para o Spectrum. Observe, entretanto, que as linhas 240, 260 e 280 são necessárias para apagar as últimas posições do avião e da bomba, à medida que estes são deslocados pela tela.

A EXPLOSÃO

Vamos agora à parte mais importante do nosso exercício de programação: a explosão. É mais fácil vê-la na tela do que tentar descrevê-la.

Portanto, antes de entrar o restante do programa, digite as linhas abaixo:

```
1000 FOR n=88 TO 80 STEP -1
1010 PRINT AT 10,15;CHR$ 150
1020 POKE 23675,n
1030 PAUSE 50
1040 NEXT n
1050 STOP
```

Mas, espere: não digite o comando **RUN**, ainda. Você poderia destruir a parte do programa que já foi digitada. Ao invés disso, digite **RUN 1000**, para começar a executar só essa parte do programa. Você verá então uma letra G aparecer na tela e depois esvanecer-se gradualmente, até ser substituída pela letra F. Isso acontece porque o laço **FOR...NEXT**, que vai da linha 1000 à 1040, diminui de um em um o valor armazenado na locação de memória 23675, que é o ponto inicial para o seu bloco gráfico UDG. Assim, a letra exibida na tela retrocede gradualmente no alfabeto.

O programa de explosão usa um truque semelhante. Para fins de segurança de seu programa, digite:

```
POKE 23675,88
```

que restaura o valor original do número armazenado na locação de memória

indicada. Em seguida, apague todas as linhas de 1000 a 1050 e digite o restante do programa de bombardeio:

```
 90 POKE 23675,88
100 FOR n=0 TO 31: READ a:
POKE USR "a"+n,a: NEXT n
300 FOR e=88 TO 80 STEP -1
310 POKE 23675,e
320 PRINT  INK 2;AT 19,12;CHR$
145 ;CHR$ 147;CHR$ 145: PAUSE
6
330 PRINT  INK 2;AT 19,12;CHR$
147 ;CHR$ 145;CHR$ 147: PAUSE
6
340 NEXT e
400 POKE 23675,88
500 GOTO 210
8000 DATA 0,0,0,0,0,0,0,0,2,128
,25,126,126,255,255,255
9000 DATA 0,0,0,0,0,0,0,0,4,33,
144,66,231,255,255,255
```

As linhas 100, 8000 e 9000 definem os UDG que você precisa para explosão. A linha 100 lê os dados contidos nas linhas 8000 e 9000, com os códigos numéricos das partes dos gráficos e os coloca na memória através do **POKE.** As linhas 300 e 340 usam a técnica descrita anteriormente para retirar uma imagem e substituí-la por outra.

Existe, entretanto, uma diferença muito importante. Os UDG que criam a explosão são baseados nos caracteres gráficos B e D — respectivamente, **CHR$ 145** e **147.** À medida que eles forem desaparecendo da tela, você não vai querer que eles sejam substituídos por outras letras do alfabeto. Assim, os primeiros oitos elementos de cada declaração **DATA,** representando os caracteres A e C — **CHR$ 144** e **146,** respectivamente — são todos zeros.

Deste modo, ao invés de obter A e C substituindo B e D na tela, você obtém espaços em branco. Se você quiser ver como isto funciona em "câmara lenta", inclua a linha abaixo (não esqueça de retirá-la, depois):

```
335 PAUSE 100
```

A linha 400 é necessária, evidentemente, para restaurar o valor original da memória 23675, que era 88, antes que o programa se repita de novo.

T

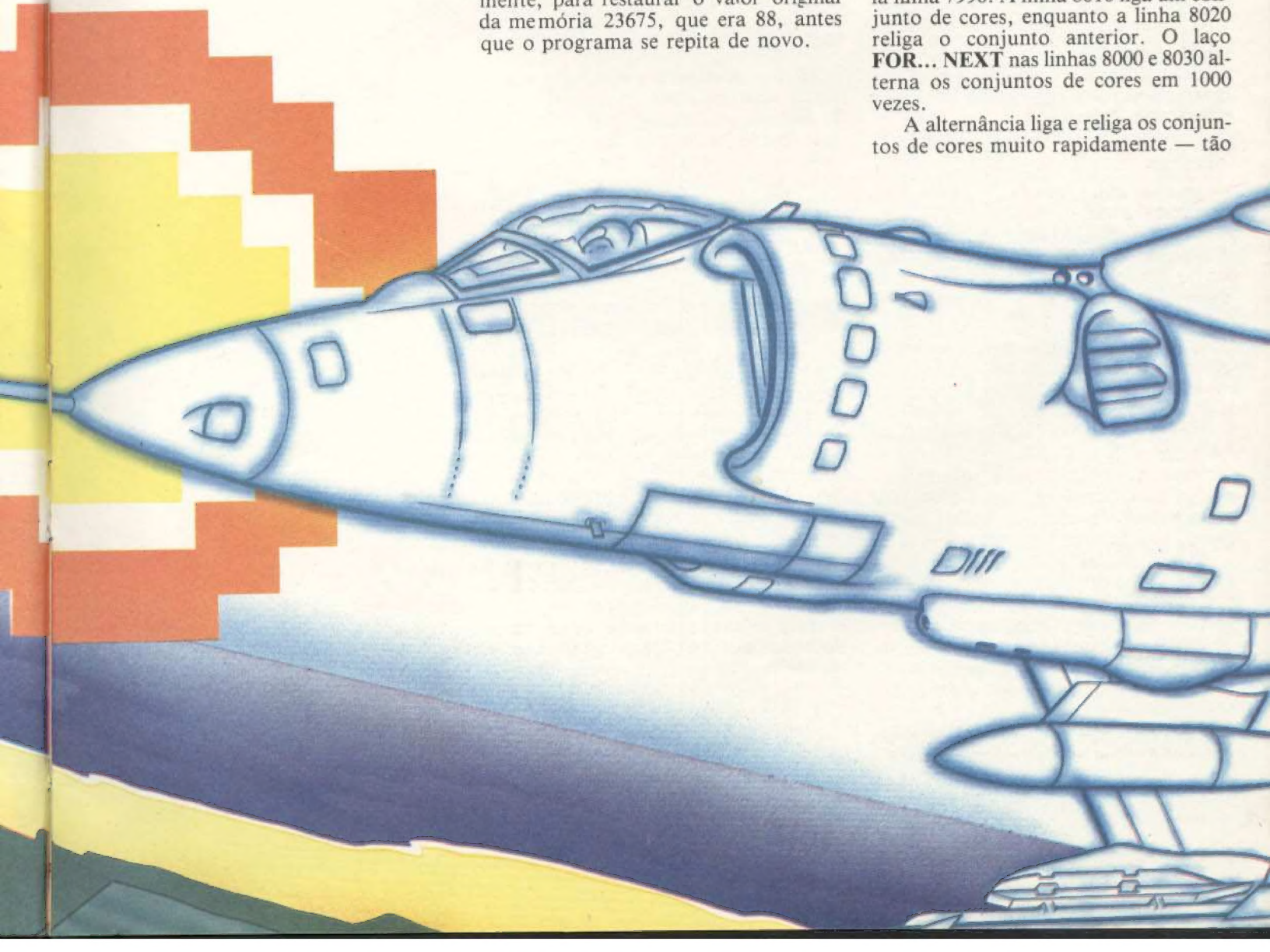
Os computadores compatíveis com essa linha têm excelentes recursos para a obtenção de efeitos visuais dramáticos, usando programas bastante curtos. Tente digitar e executar o programa abaixo:

```
7980 PMODE 3,1
7990 PCLS
8000 FOR F=1 TO 1000
8010 SCREEN 1,0
8020 SCREEN 1,1
8030 NEXT F
8040 CLS
```

Observe que o programa coloca faixas coloridas que migram através da tela; esse efeito é causado pela alternância super-rápida dos diferentes conjuntos de cores no computador. Ele pode ser usado, entre outras coisas, para assimilar o fim de uma fase do jogo, como uma blindagem que foi penetrada.

O modo gráfico é definido pela linha 7980, e a seguir a tela gráfica é limpa pela linha 7990. A linha 8010 liga um conjunto de cores, enquanto a linha 8020 religa o conjunto anterior. O laço **FOR... NEXT** nas linhas 8000 e 8030 alterna os conjuntos de cores em 1000 vezes.

A alternância liga e religa os conjuntos de cores muito rapidamente — tão



rapidamente, na verdade, que o aparelho de TV nunca tem tempo de mudar as cores na tela completamente antes de entrar em ação o outro conjunto. Como conseqüência, apenas uma faixa estreita da tela é mudada para cada cor.

Se você pretende utilizar esse programa na forma de uma sub-rotina dentro de outro jogo (no qual funcionará apenas com um dos comandos **PMODE** descritos na página 144) ou com o programa de batalha espacial apresentado a seguir, terá que fazer algumas alterações. Primeiro, apague as linhas 7980 e 7990; em seguida, adicione a linha:

```
8500 RETURN
```

Veja agora o programa a seguir:

```
10 PMODE 1,1
20 DIM A(3),B(3),M(3)
200 FOR K=1536 TO 2016 STEP 32
210 READ A,B: POKE K,A:POKE K+1
,B
220 NEXT
230 GET(0,0)-(15,15),A,G
240 GET(0,16)-(15,31),M,G
250 PCLS
260 MX=120:MY=191:PX=120:PY=20
270 PUT (PX,PY)-(PX+15,PY+15),A
,PSET
280 SCREEN 1,0
290 PUT (MX,MY)-(MX+15,MY+15),B
,PSET
300 MY=MY-4
310 IF MY<36 THEN GOSUB 8000:GO
TO 260
320 PUT (MX,MY)-(MX+15,MY+15),M
,PSET
330 GOTO 290
9000 DATA 252,63,3,192,15,240,6
1,124,58,172,245,95,213,87,213,
87
9010 DATA 0,128,0,128,0,128,2,1
60,10,168,0,192,3,240,15,60
```

O programa mostra uma nave extraterrestre na iminência de ser atingida por um míssil. No momento que este alcança o alvo, entra em ação a sub-rotina definida anteriormente, que move faixas coloridas pela tela.

Quando usado com a sub-rotina, ele faz faixas com qualquer **PMODE** que se queira, de modo que pode ser adicionado ao final de um programa, sem que seja necessário alterar as linhas 8000 a 8040.

Eis aqui uma adaptação do programa para fazer faixas que resulta em um efeito visual mais elaborado, embora os gráficos na tela não permaneçam intocados como anteriormente. Você pode digitá-lo e executá-lo do jeito que está, ou incorporá-lo a um outro programa — como o da animação da nave espacial — na forma de sub-rotina. Neste caso, você precisará alterar o programa, apagando a linha 7990 e adicionando ao final uma linha com comando **RETURN** como foi feito anteriormente.

```
7990 PMODE 3,1
8000 FOR F=1 TO 3
8010 FOR K=0 TO 1
8020 SCREEN 1,K
8030 FOR J=1 TO 4
8040 PCLS J
8050 NEXT J
8060 NEXT K
8070 NEXT F
8080 CLS
```

Sob a forma de sub-rotina, ele funcionará em qualquer um dos modos gráficos com conjuntos de duas cores (**PMODE 0, 2 e 4**) e de quatro cores (**PMODE 1 e 3**). Além de mudar os conjuntos de cores, a rotina também limpa a tela.

As linhas 8030 a 8050 são um laço **FOR... NEXT** que colore a tela com as cores disponíveis no conjunto ativo.

Uma última variação do programa:

```
7990 PMODE 3,1
8000 FOR F=1 TO 5
8010 SCREEN 1,0
8020 FOR K=1 TO 200: NEXT K
8030 SCREEN 1,1
8040 FOR K=1 TO 200: NEXT K
8050 NEXT F
8060 CLS
```

Você pode escolher entre digitá-lo assim como está (como um programa autônomo) ou na forma de sub-rotina (digitando a linha 7990 e adicionando um comando **RETURN** ao seu final).

A tela piscará rapidamente, pois as linhas 8020 e 8040 inserem pequenos retardos de tempo, dando oportunidade suficiente para a mudança de cor entre um conjunto e outro.

CHAMAS

Podemos obter efeitos visuais ainda melhores através da programação de gráficos animados. Eis aqui uma idéia para fazer a animação gráfica das chamas de um incêndio, que você poderia, por exemplo, superpor a um alvo atingido por uma bomba. O programa foi escrito para o **PMODE 1**, e não pode ser usado, do jeito que está, com jogos em outros **PMODE**. Digite-o e execute-o:

```
10 PMODE 1,1
20 DIM B(3),E1(3),E2(3)
30 FOR K=1536 TO 2016 STEP 32
40 READ A,B:POKE K,A:POKE K+1,B
50 NEXT
60 GET(0,0)-(15,15),E1,G
70 GET(0,16)-(15,31),E2,G
80 SCREEN 1,0
250 PCLS:HX=124:HY=146
8000 FOR N=0 TO 15
8010 PUT (HX,HY+N)-(HX+15,HY+15
),E1,PSET
8020 FOR K=1 TO 100:NEXT
8030 PUT (HX,HY+N)-(HX+15,HY+15
),E2,PSET
8040 FOR K=1 TO 100:NEXT
8050 PUT (HX,HY+N)-(HX+15,HY+15
),B,PSET
8060 NEXT
9000 DATA 0,12,192,0,3,195,63,2
52,63,252,255,255,255,255,255,2
55
9010 DATA 0,48,12,3,195,0,48,12
,252,63,255,255,255,255,255,255
```

Dois conjuntos de chamas são colocados na tela pelo comando **PUT**.

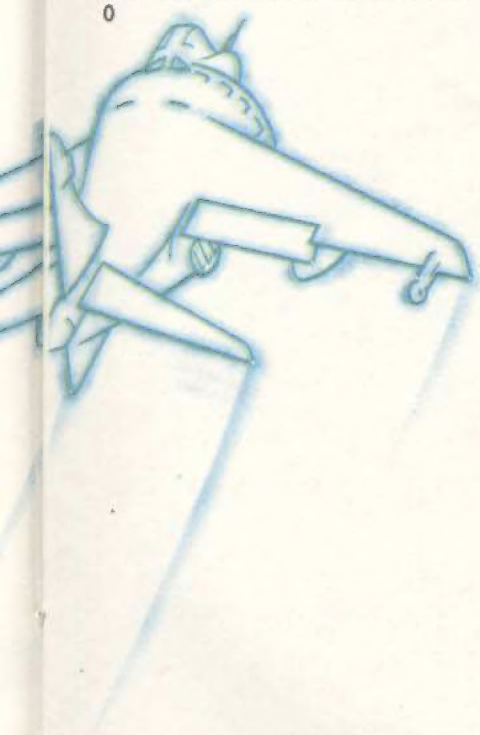
Os dados para os conjuntos de chamas estão nas declarações **DATA** das linhas 9000 e 9010. Eles são colocados com comandos **POKE** na tela, com as linhas 30 e 50. O programa está escrito em uma modalidade gráfica de quatro cores. As linhas 60 e 70 capturam, através do comando **GET**, as formas gráficas na tela e as armazenam nos conjuntos E1 e E2, dimensionados na linha 20.

As linhas 8000 e 8060 animam as chamas. Quando o programa passa pelo laço **FOR...NEXT**, os conjuntos E1 e E2 surgem na tela, e apagam quando se coloca sobre eles o conjunto de espaços

em branco, B, com **PUT**. O topo das chamas é abaixado um pouco de cada vez, pelo + **N** nas linhas com comando **PUT**.

Você pode usar esse programa como sub-rotina do programa abaixo, que simula um bombardeio, mas não se esqueça de alterar a linha 20 para como ela aparece no programa: edite-a com o comando **EDIT** ou digite-a novamente. Apague a linha 80, digitando o número 80, seguido de <**ENTER**>.

```
20 DIM A(3),B(3),H(3),E1(3),E2(
3)
200 FOR K=1536 TO 2016 STEP 32
210 READ A,B:POKE K,A:POKE K+1,
B
220 NEXT
230 GET(0,0)-(15,15),A,G
240 GET(0,16)-(15,31),H,G
250 PCLS:LINE(0,,163)-(255,191),
PSET,BF
260 HX=124:HY=146:PX=0:PY=40:B=
0
270 PUT (HX,HY)-(HX+15,HY+15),H
,PSET
280 SCREEN 1,0
290 PUT (PX,PY)-(PX+15,PY+15),B
,PSET
300 PX=PX+4
310 PUT(PX,PY)-(PX+15,PY+15),A,
PSET
320 IF PX=20 THEN B=1:BX=PX+8:B
Y=PY+8
330 IF B=1 THEN PRESET(BX,BY):P
RESET(BX+2,BY):BX=BX+2:BY=BY+2:
PSET(BX,BY,4):PSET(BX+2,BY,4)
340 IF BY=148 THEN GOSUB 8000:B
Y=0:GOTO 250
350 GOTO 290
8070 RETURN
9020 DATA 0,0,2,0,130,128,162,1
60,170,170,162,160,130,128,2,0
9030 DATA 0,3,12,51,60,243,255,
255,255,255,85,85,86,149,86,149
```



Eis aqui duas versões que permitirão usar a sub-rotina em jogos programados em outras modalidades gráficas (**PMODE**). A primeira é um programa que deve ser usado com as **PMODE 3** e **4**; mude apenas o número apropriado.

```
10 PMODE 3,1
20 DIM B(6),E1(6),E2(6)
30 FOR K=1536 TO 2496 STEP 64
40 READ A,B:POKE K,A:POKE K+1,B
45 POKE K+32,A:POKE K+33,B
50 NEXT
60 GET (0,0)-(15,15),E1,G
70 GET (0,16)-(15,31),E2,G
80 SCREEN 1,0
250 PCLS:HX=124:HY=146
8000 FOR N=0 TO 15
8010 PUT(HX,HY+N)-(HX+15,HY+15)
,E1,PSET
8020 FOR K=1 TO 100: NEXT
8030 PUT (HX,HY+N)-(HX+15,HY+15
),E2,PSET
8040 FOR K=1 TO 100:NEXT
8050 PUT(HX,HY+N)-(HX+15,HY+15)
,B,PSET
8060 NEXT
9000 DATA 0,12,192,0,3,195,63,2
52,63,252,255,255,255,255,255,2
55
9010 DATA 0,48,12,3,195,0,48,12
,252,63,255,255,255,255,255,255
```

Em segundo lugar, temos um programa adequado para uso com **PMODE 2**:

```
10 PMODE 2,1
20 DIM B(6),E1(6),E2(6)
30 FOR K=1536 TO 2496 STEP 32
40 READ A: POKE K,A: POKE K+16,
A
50 NEXT
60 GET(0,0)-(15,15),E1,G
70 GET(0,16)-(15,31),E2,G
80 SCREEN 1,0
250 PCLS:HX=124:HY=146
8000 FOR N=0 TO 15
8010 PUT(HX,HY+N)-(HX+15,HY+15)
,E1,PSET
8020 FOR K=1 TO 100:NEXT
8030 PUT (HX,HY+N)-(HX+15,HY+15
),E2,PSET
8040 FOR K=1 TO 100: NEXT
8050 PUT(HX,HY+N)-(HX+15,HY+15)
,B,PSET
8060 NEXT
9000 DATA 2,128,25,126,126,255,
255,255
9010 DATA 4,33,144,66,231,255,2
55,255
9020 DATA 0,12,192,0,3,195,63,2
52,63,252,255,255,255,255,255,2
55
```

MSX

Os computadores compatíveis com a linha MSX têm excelentes recursos (tais como os *sprites*), para se conseguir efeitos visuais dramáticos, usando-se programas bastante curtos. Tente digitar e executar o programa abaixo:

```
7990 SCREEN 0:KEY OFF
8000 FOR F=1 TO 300
8010 COLOR 10,10,10
8020 COLOR 4,4,4
8030 NEXT F
8040 COLOR 15,4,4
```

O modo de texto da tela é definido pela linha 7990 (**SCREEN 0**). A linha 8010 estabelece uma cor de fundo (10) para a tela, com a mesma cor de moldura e de frente (por isso todos os números são iguais); a linha 8020 especifica um segundo conjunto de cores para a tela. O laço **FOR...NEXT** nas linhas 8000 e 8030 alterna os conjuntos de cores trezentas vezes. A linha 8040 volta tudo à situação original de cor da tela, antes de interromper o programa (se você interrompê-lo usando as teclas <**CTRL**> <**STOP**>, antes de terminar o laço de repetição, terá que digitar pelo teclado, ou com a tecla F1, o comando existente na linha 8040).

Como a alternância das cores é muito rápida, apenas uma faixa estreita da tela é mudada para cada cor. Esse efeito só pode ser conseguido, no MSX, com a tela de texto. Podemos introduzi-lo como fase final de um programa de animação gráfica. Acrescente as seguintes linhas ao programa:

```
10 SCREEN 2,2
200 FOR K=1 TO 24
210 READ A:READ B
220 A$=A$+CHR$(A):B$=B$+CHR$(B)
230 NEXT
240 SPRITE$(2)=A$:SPRITE$(1)=B$
250 MX=120:MY=191
260 PX=120:PY=20
270 PUT SPRITE 2,(PX,PY)
280 PUT SPRITE 1,(MX,MY)
290 PUT SPRITE 1,(MX,MY)
300 MY=MY-4
310 IF MY<36 THEN GOTO 7990
320 PUT SPRITE 1,(MX,MY)
330 GOTO 290
```

```
8050 RUN
9000 DATA 252,0,3,0,15,0,61,2,5
8,10,24,5,0,213,3,213,15
9010 DATA 0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0
9020 DATA 63,128,192,128,240,12
8,124,160,172,168,95,192,87,240
,87,60
```

Os dois objetos a serem movimentados na tela (um míssil e uma nave) são definidos pelos códigos gráficos das linhas 9000 a 9020, montados nos sprites 1 e 2. As linhas 270 e 290 colocam esses sprites na tela, animando o sprite 1 (o míssil) através do ciclo repetido entre 290 e 330.

Quando o míssil atinge a nave (linha 310), o programa de efeito visual de faixas começando em 7990 é acionado. A linha 8050 manda executar o programa principal desde o começo, em virtude da necessidade de redefinição da tela gráfica na linha 10.

Em seguida, temos uma adaptação do programa para fazer faixas. Você pode digitá-lo e executá-lo como está, ou adicioná-lo a outro programa, como o da animação da nave espacial.

```
7990 SCREEN 0:KEY OFF
8000 FOR F=1 TO 15
8005 FOR K=1 TO 15
8010 COLOR F,K,K
8020 COLOR K,F,F
8025 NEXT K
8030 NEXT F
8040 COLOR 15,4,4
8050 RUN
```

As linhas 8000 a 8030 formam dois laços **FOR...NEXT** que colorem a tela com cores disponíveis na tela do texto MSX. A última variação do programa é:

```
7990 SCREEN 0:KEY OFF
8000 FOR F=1 TO 5
8010 COLOR 4,10,10
8015 FOR K=1 TO 150:NEXT K
8020 COLOR 10,4,4
8025 FOR K=1 TO 150:NEXT K
8030 NEXT F
8040 COLOR 15,4,4
8050 RUN
```

A tela pisca rapidamente, em cores porque as linhas 8015 e 8025 inserem pequenos retardos de tempo, dando oportunidade para o vídeo mudar de cor entre um conjunto e outro.

FOGO!

Efeitos visuais ainda melhores podem ser obtidos através da programação de sprites nos gráficos animados. Eis aqui uma idéia para fazer a animação gráfica de um incêndio.

O programa foi escrito para a tela gráfica **SCREEN 2**. Esse comando também é usado para definir o tamanho do sprite. Digite o comando **NEW** e entre as seguintes linhas:

```
10 SCREEN 2,2
30 FOR K=1 TO 32
40 READ A:READ B
45 A$=A$+CHR$(A):B$=B$+CHR$(B)
50 NEXT
60 SPRITE$(2)=A$
70 SPRITE$(1)=B$
75 SPRITE$(0)=STRING$(32,255)
250 HX=124:HY=146
8000 PUT SPRITE 0,(HX,HY+16),12
8005 FOR N=0 TO 15
8010 PUT SPRITE 2,(HX,HY+N),6
8020 FOR K=1 TO 100:NEXT
8025 PUT SPRITE 2,(HX,HY+16)
8030 PUT SPRITE 1,(HX,HY+N),6
8040 FOR K=1 TO 100:NEXT
8060 NEXT
9000 DATA 0,0,192,12,3,195,63,4
8,63,252,255,255,255,255,255,25
5
9010 DATA 255,255,255,255,255,2
55,255,255,255,255,255,255,255,
255,255,255
9020 DATA 12,48,0,3,195,0,252,1
2,252,63,255,255,255,255,255,25
5
9030 DATA 255,255,255,255,255,2
55,255,255,255,255,255,255,255,
255,255,255
```

Dois conjuntos de chamas, armazenados em sprites de tamanho 2 (ou seja, 16 por 16 pixels), são colocados na tela pelo comando **PUT**. Os dados para os conjuntos de chamas estão nas declarações **DATA** das linhas 9000 a 9030. Eles são lidos pelas declarações **DATA** na linha 40 e colocados nas variáveis **A$** e **B$**, na linha 45. Após isso, os sprites números 1 e 2 recebem o conteúdo desses cordões. Um terceiro sprite (um bloco gráfico uniformemente cheio) também é definido na linha 75. A cor que esse bloco de sprite vai assumir ao ser colocado na tela é determinada pelo comando **PUT** na linha 8000 (no caso, azul, que é o código 4).

As linhas 8000 a 8060 animam as chamas. Cada vez que o programa passa pelo laço **FOR...NEXT**, os dois sprites 1 e 2 são colocados na tela. O topo das chamas é abaixado um pouco de cada vez, pelo **+N** nas linhas com o comando **PUT**, de modo que dá a impressão de diminuição do fogo.

Esse programa pode ser usado como uma sub-rotina do programa a seguir, que simula um bombardeio: um prédio é alvejado. Adicione as linhas:

```
200 FOR K=1 TO 32
210 READ A:READ B
215 D$=D$+CHR$(A):E$=E$+CHR$(B)
220 NEXT
230 SPRITE$(3)=D$
240 SPRITE$(4)=E$
250 LINE(0,163)-(255,191),12,BF
260 HX=124:HY=146:PX=0:PY=40:B=
0
270 PUT SPRITE 4,(HX,HY),9
300 PX=PX+4
310 PUT SPRITE 3,(PX,PY),14
320 IF PX=20 THEN B=1:BX=PX+8:B
Y=PY+8
330 IF B=1 THEN PRESET (BX,BY):
PRESET(BX+2,BY):BX=BX+2:BY=BY+2
:PSET(BX,BY),1:PSET(BX+2,BY),1
340 IF BY=148 THEN PUT SPRITE 4
,(HX,HY+16):PRESET (BX,BY):PRES
ET (BX+2,BY):GOSUB 8000:BY=0:GO
TO 250
350 GOTO 300
8070 RETURN
9040 DATA 0,0,0,0,2,12,2,12,130
,60,130,60,162,255,162,255
9050 DATA 170,255,170,255,162,8
5,162,85,130,86,130,86,2,86,2,8
6
9060 DATA 0,3,0,3,0,51,0,51,128
,243,128,243,160,255,160,255
9070 DATA 170,255,170,255,160,8
5,160,85,128,149,128,149,0,149,
0,149
```

Nesse trecho suplementar, mais dois sprites, 3 e 4, são definidos pelas linhas 200 a 240, a partir dos códigos gráficos armazenados nas declarações **DATA** nas linhas 9040 a 9070.

A linha 250 traça um retângulo de cor verde (o chão gramado); a linha 270 coloca dentro dele o desenho da casa. A animação gráfica do avião e da bomba caindo é realizada pelas linhas 310 e 350. Na linha 340 é chamada a sub-rotina definida a partir de 8000, que recebe um **RETURN** na linha 8070.

Não é tão fácil criar efeitos visuais no Apple se o compararmos com outros micros. Ao invés de representar os desenhos através de números binários convertidos para decimais e colocados na memória de vídeo (''1'' representaria ponto aceso, e ''O'', ponto apagado), temos que criar uma tabela que pode ter até 256 figuras, e uma vez que esteja num lugar apropriado da memória pode-se desenhar e mover as figuras com o comando **DRAW**, sendo a cor definida por **HCOLOR**. O programa a seguir move uma chama na base da tela.

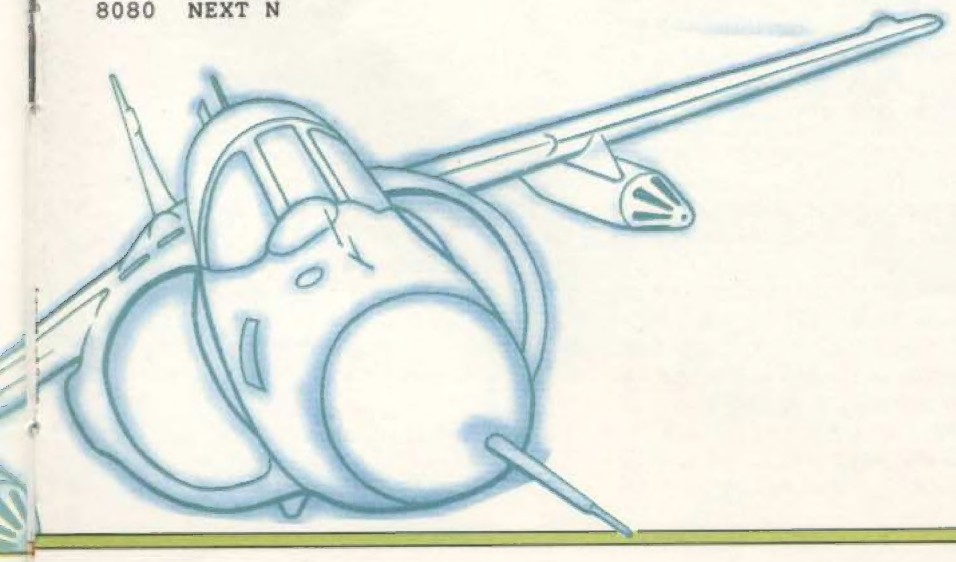
```
10  HOME : HGR : SCALE= 1: ROT=
 64
20  POKE 232,0: POKE 233,3
30  FOR K = 768 TO 868
40  READ A: POKE K,A
50  NEXT
250 HX = 124:HY = 146
8000  FOR N = 1 TO 15
8010  HCOLOR= 5
8020  FOR K = 1 TO 3
8030  DRAW 1 AT HX,HY + N: HCOL
OR= 0
8040  DRAW 1 AT HX,HY + N: HCOL
OR= 5
8050  DRAW 1 AT HX + 3,HY + N +
 2: HCOLOR= 0
8060  DRAW 1 AT HX + 3,HY + N +
 2: HCOLOR= 5
8070  NEXT K
8080  NEXT N
8090  TEXT
9000  DATA  3,0,8,0,64,0,0,1,15
0,18,54,37,36,36,4,11,54,54,54,
46,36
9010  DATA  36,11,54,54,46,36,4
,100,100,11,54,54,54,37,36,4,10
0,100,11,54
9020  DATA  54,54,46,36,100,100
,100,11,22,54,54,37,36,100,100,
11,54,54,54,37
9030  DATA  36,4,0,45,56,255,21
9,35,109,73,45,37,63,63,63,63,3
9,45,45,45
9040  DATA  45,45,46,44,46,45,3
7,63,231,219,63,255,219,39,64,7
3,44,46,5,0
```

O mesmo incêndio é desenhado e apagado rapidamente em duas posições diferentes, dando a impressão de fogo. As chamas desaparecem lentamente na parte inferior do vídeo. O programa funciona da seguinte maneira:

A tabela de figuras está nas linhas **DATA** (9000 a 9040). Ela é colocada na memória pela repetição do comando **POKE** da linha 40, devida ao laço **FOR...NEXT** das linhas 30 e 50. A linha 20 informa onde está a tabela de figuras na memória. A linha 10 limpa a tela, estabelece o modo gráfico de alta resolução, fixa o tamanho (**SCALE**) e a orientação (**ROT**) da figura.

As linhas 8000 a 8060 animam as chamas e as fazem desaparecer. Cada vez que o programa repete as linhas que ficam entre **FOR** e **NEXT**, as chamas são desenhadas e apagadas em duas posições diferentes da tela. Elas vão baixando lentamente devido ao **+N** nas linhas **HPLOT**, de forma que o fogo parece acabar. Isso acontece porque o desenho invade o espaço reservado para texto na parte inferior da tela, onde podemos colocar dados gráficos sem que eles apareçam. Para entender melhor, experimente introduzir a linha:

```
15 POKE -16302,0
```



Essa linha elimina o espaço reservado para textos da parte inferior da tela, tornando visíveis quaisquer gráficos ali colocados.. Rodando o programa com essa modificação, você verá as chamas ocuparem posições cada vez mais baixas sem desaparecer. Não se esqueça de apagar essa linha digitando 15 e <**RETURN**>. Nosso incêndio ficará mais interessante se for produzido por uma sub-rotina dentro de um jogo, como no exemplo a seguir. Após acrescentar essas linhas, você verá uma pequena casa ser bombardeada e pegar fogo. Não se esqueça de apagar a linha 15.

```
55 I = 0
60  FOR J = 8674 TO 15810
    STEP 10 24
70  READ A(I),B(I)
80 I = I + 1
100  NEXT
190 I = 0
200  FOR J = 8674 TO 15810
     STEP 1 024
210  POKE J,B(I): POKE J +
     1,A(I)
215 I = I + 1
220  NEXT
235  FOR K = 1 TO 1000: NEXT
250 HX = 139:HY = 153:PX = 259:
      PY = 40:B = 0
290 PX = PX - 4
300  HCOLOR= 6: DRAW 2 AT PX,PY
310  HCOLOR= 0: DRAW 2 AT PX,PY
320  IF PX = 239 THEN B =
     1:BX = PX:BY = PY + 8
330  IF B = 1 THEN  HCOLOR= 0:
     HPLOT BX,BY: HPLOT BX -
     2,BY:BX = BX - 2:BY =
     BY + 2: HCOLOR= 3: HPLOT
     BX,BY: HPLOT BX - 2,BY
340  IF BY = 152 THEN  GOSUB
     8000 :BY = 0: GOTO 190
350  GOTO 290
8090  RETURN
9050  DATA   0,3,12,51,60,243,
      255,255,255,255,85,85,86,
      149,86,149
```

A casa é desenhada pelos comandos **POKE** da linha 210. O padrão do desenho é obtido da linha 9050 e guardado nas variáveis indexadas **A** e **B**. Programar desenhos dessa maneira é mais fácil que definir tabelas, mas quando se deseja movimento e velocidade (como nas chamas e no avião) é preciso usar as tabelas e o comando **DRAW**.

O formato do avião já estava na tabela do programa anterior. As chamas são a figura 1 da tabela, e o avião é a figura 2. Essas figuras são desenhadas respectivamente pelos comandos **DRAW 1** e **DRAW 2**.

Para parar o programa aperte simultaneamente <**CTPL**> e C. Mesmo que não apareça nada na tela, digite **TEXT** e aperte <**RETURN**> e tudo voltará ao normal.

OS COMANDOS READ E DATA

Se você pretende evitar a digitação de programas longos e cansativos, convém preparar seu computador para a leitura de dados armazenados internamente em declarações DATA.

As variáveis são o grande responsável pela versatilidade do computador. Atribuir valores a elas é, a maior parte das vezes, simples e direto: basta, por exemplo, dizer **LET X = 5.** Mas há ocasiões em que a quantidade de informações que se deseja utilizar em um programa é muito grande. Nesses momentos, é preciso recorrer às declarações **DATA**, e aos comandos **READ** e **RESTORE** (esses comandos não estão disponíveis no BASIC de alguns micros).

A palavra **DATA** (dados, em inglês) é utilizada para muitas coisas. Assim, tudo o que é fornecido ("dado" pelo programador ou pelo usuário) ao computador é incluído nessa categoria. Nesse artigo, porém, vamos usar a expressão dados apenas em seu sentido mais restrito, ou seja, informações incluídas em um programa por meio da declaração **DATA.**

A declaração **INPUT** serve para atribuir valores a uma variável. Mas ela é útil somente enquanto a informação estiver sendo fornecida ao computador cada vez que o programa for rodado.

Existem valores fixos, porém, que não precisam ser entrados ou alterados pelo operador, e podem, assim, ser incorporados permanentemente ao programa. Neste caso, entra em cena o segundo método de atribuição: a declaração **LET.** Mas, quando as informações são em grande quantidade, o emprego de comandos **LET** torna-se cansativo.

Veja a seguir um exemplo onde uma lista fixa de cabeçalhos é utilizada para imprimir um documento:

```
10 LET A$="DIA"
20 LET B$="SEMANA"
30 LET C$="MES"
40 LET C$="ANO"
50 PRINT A$,B$,C$,D$
```

Veja como **DATA** pode ser utilizada com o mesmo objetivo e com uma redução no número de linhas necessárias:

```
10 READ A$,B$,C$,D$
20 PRINT A$,B$,C$,D$
100 DATA DIA,SEMANA,MES,ANO
```

Em programas curtos não há quase diferença entre usar **LET** ou **DATA.** Agora, se você quiser trabalhar com cinqüenta cabeçalhos, em vez de quatro, o primeiro programa precisará de 46 declarações **LET** a mais, enquanto o segundo exigirá apenas uma ou duas linhas a mais.

Nos compatíveis com o Spectrum (como o TK-90X), as palavras **DATA** devem estar sempre entre aspas. Nos outros modelos as aspas são dispensáveis. A linha 100 passaria a ser escrita assim:

```
100 DATA  "DIA","SEMANA","MES",
    "ANO"
```

COMO FUNCIONAM READ E DATA

Quando o computador se depara com uma instrução **READ**, ele passa a procurar em todo o programa onde se encontra a primeira declaração **DATA**. Então ele atribui o primeiro item de **DATA** à variável correspondente no comando **READ**. No programa acima, o cordão "DIA", na declaração **DATA**, é atribuído à variável **A$**, "SEMANA" é atribuído a **B$**, etc. O computador ignora qualquer declaração **DATA** a menos que uma declaração **READ** faça com que ele a leia.

```
10 DATA ALEMANHA
20 READ A$,B$
30 DATA BRASIL,ITALIA,ESPANHA
40 READ C$,D$
```

É mais fácil ler o programa quando todos os **DATA** estão reunidos em um mesmo lugar. E existe uma regra inabalável: os comandos **DATA** devem aparecer na ordem em que o computador vai lê-los, por meio da instrução **READ.**

DIFERENTES TIPOS DE DATA

Além de cordões alfanuméricos, as declarações **DATA** podem conter números. A linha Sinclair Spectrum também

- COMO FUNCIONAM AS DECLARAÇÕES **READ** E **DATA**
- DIFERENTES TIPOS DE **DATA**
- UTILIZE **DATA** PARA FAZER UMA LISTA TELEFÔNICA
- A DECLARAÇÃO **RESTORE**
- APLICAÇÕES PARA LISTAS DE **DATA**
- COMO DESENHAR GRÁFICOS SIMPLES A PARTIR DE **DATA**

permite que você utilize variáveis e funções dentro de declarações **DATA.**

```
DATA "RECIFE",256,a*5,SQR(num)
```

No comando **READ** as variáveis devem ser colocadas na mesma ordem que os itens relativos a **DATA**. A linha **DATA** acima pode ser lida com o comando:

```
READ A$,N,X,Y
```

A primeira variável é uma variável *alfanumérica*, para coincidir com o primeiro item de **DATA.** Mas os três itens seguintes são variáveis *numéricas.*

É fácil cometer erros quando se usa **DATA** em um programa. Os principais problemas ocorrem quando não se possui um número suficiente de itens ou quando se tenta ler com **READ** uma variável de outro tipo, em **DATA.** Se a linha **DATA** acima tivesse sido digitada incorretamente, tal como **DATA 256, RECIFE, a★5,** então **A$** receberia o valor 256! Embora errada, ela não seria rejeitada (a não ser no Spectrum) pois o computador interpreta o valor numérico 256, neste caso, como o cordão alfanumérico "256".

Em contrapartida, se o computador tentasse ler o cordão "RECIFE" e o atribuísse à variável númerica N, poderia perceber o erro e responder com uma mensagem tal como: *"tipo inadequado"* (erro TM, ou TYPE MISMATCH), ou *"dado incorreto"* (erro BAD DATA, ou INVALID DATA, dependendo do computador); ou poderia supor que "RECIFE" é um nome de variável não definido anteriormente; sua resposta, neste caso, seria uma mensagem de erro como: *"variável não encontrada".*

Outro erro comum é colocar itens a menos em **DATA.**

UMA LISTA TELEFÔNICA

Eis aqui um programa que utiliza um laço para a leitura interna de dados, através das declarações **READ** e **DATA.** É um programa muito simples para elaborar uma lista telefônica particular.

O programa abaixo roda apenas no TRS-Color. Para utilizá-lo nos micros TRS-80, modifique os números do **PRINT** nas linhas 50 e 60, para 640.

```
5 CLS
10 PRINT @70,"DIRETORIO TELEFON
ICO"
12 PRINT
20 INPUT "DIGITE O NOME";R$
30 FOR J=1 TO 5
40 READ N$,T$
50 IF N$=R$ THEN PRINT @320,"O
NUMERO DE ";R$;" E: ";T$:END
60 IF N$="FIM" THEN PRINT @320,
R$;" NAO ESTA NA LISTA"
70 NEXT J
500 DATA ALCINO,384-4276,JUNIOR
,997-4036,RODRIGO,52-5114,GERTR
UDES,5666-99994,FIM,FIM
```

```
 5 CLS
10 PRINT AT 2,6;"GUIA TELEFON
ICO"
15 RESTORE
20 INPUT "Digite o nome",R$
30 FOR J=1 TO 5
40 READ N$,T$
50 IF N$=R$ THEN  PRINT AT 10
,1;"O numero de ";N$;" e ";T$
: GOTO 80
60 IF N$="FIM" THEN  PRINT AT
10,3;R$;" NAO ESTA NA LISTA"
70 NEXT J
80 PRINT AT 12,0;"Quer outro
numero (S/N)?"
90 PAUSE 0
100 IF INKEY$="S" THEN  GOTO 5
110 IF INKEY$="N" THEN  GOTO
2000
500 DATA "ZULEICA","22-6400","
MARIA", "33-7237","MARCELO","3
4-4009","RICARDO","13-71824","
FIM","FIM"
```

```
10 LOCATE 12,3:PRINT"Lista tele
fônica"
12 PRINT:PRINT
15 RESTORE
20 INPUT"Digite o nome";R$
30 FORJ=1TO5
40 READ N$,T$
50 IFN$=R$ THEN LOCATE 5,15:PRI
NT"O número de ";N$;" é ";T$:GO
TO 80
60 IFN$="END" THEN LOCATE 5,15:
PRINTR$;" não está na lista!"
70 NEXTJ
500 DATA ESTELA,321054,JOSÉ,529
884,CELI,326747,FERNANDO,511934
,END,END
```

```
5  HOME
10  VTAB 3: HTAB 12: PRINT "Lis
ta telefonica"
12  PRINT : PRINT
20  INPUT "Digite o nome: ";R$
30  FOR J = 1 TO 5
40  READ N$,T$
50  IF N$ = R$ THEN  VTAB 15: H
TAB 5: PRINT "O numero de ";N$;
" e ";T$: END
60  IF N$ = "END" THEN  VTAB 15
: HTAB 5: PRINT R$;" nao esta n
a lista!"
70  NEXT J
500  DATA  FERNANDO, 510536, MA
RILIA, 543286,JOAQUIM,28999,ROB
ERTA,719852,END,END
```

Você pode utilizar os nomes e os números dos telefones dos seus amigos na linha **DATA** e colocar tantos itens **DATA** quanto desejar; mas não se esqueça de ajustar o contador de laços na linha 30.

A lista de itens em **DATA** é finalizada com as palavras FIM, FIM, para a linha 60 checar se já foi alcançado o fim da lista. Se o programa tiver lido a lista em **DATA** até o fim, isso significa que o nome não foi encontrado, e o programa imprimirá uma mensagem para dizer isto a você. FIM (ou "FIM" no Spectrum) deve ser entrado duas vezes porque a linha 40 lê dois itens **DATA** de cada vez.

Os cordões em **DATA** podem conter espaços, mas não vírgulas. Se você precisar entrar um cordão **DATA** com uma vírgula, coloque o cordão entre aspas:

```
10 READ N$,A$;B$,C$
20 DATA JOSE DOS CAMPOS,
   "AV. BRASIL,23",
   CAMPINAS,13100
```

A primeira linha do endereço deve ser escrita entre aspas por causa da vírgula depois de AV. BRASIL.

COMO USAR A DECLARAÇÃO RESTORE

Com os métodos vistos até agora, um programa poderá ler uma lista de **DATA** apenas uma vez, a menos que você o rode de novo. Para ler mais uma vez o conjunto de itens em **DATA** você deve incorporar ao seu programa a declaração **RESTORE** (*restaurar*, em inglês). Se você quiser consultar os números dos telefones dos amigos sem rodar o programa, substitua **END** ou **STOP** na linha 50 por **GOTO 80** e acrescente:

O programa listado abaixo roda apenas no TRS-Color. Para utilizá-lo em micros da linha TRS-80, modifique o número do **PRINT** na linha 80 para 832.

```
80 PRINT @416,"VOCE QUER OUTRO
NUMERO(S/N)?"
90 K$=INKEY$:IF K$="" THEN GOTO
 90
100 IF K$="S" THEN GOTO 5
110 END
```

```
 80 PRINT AT 12,0;"Quer outro
 numero (S/N)?"
 90 PAUSE 0
100 IF INKEY$="S" THEN  GOTO 5
110 IF INKEY$="N" THEN  GOTO
2000
```

```
80 LOCATE 3,21:PRINT"Você quer
outro número? (S/N)"
90 K$=INKEY$:IF K$="" THEN 90
100 IF K$="S" THEN 5
110 END
```

```
80  VTAB 22: HTAB 1: PRINT "Voc
e deseja mais algum numero? (S/
N)";
90  GET K$
100  IF K$ = "S" THEN 5
110  END
```

Mas o que acontecerá quando você rodar o programa? Na primeira vez, ele funcionará bem. Mas, se você pressionar uma tecla para outra repetição da consulta, ele falhará devido à falta de mais itens em **DATA.** Mas existe uma solução para o problema. Acrescente:

```
15 RESTORE
```

Desta vez o programa continuará funcionando toda vez que se repetir, pois o comando **RESTORE** instrui o computador a voltar ao início da lista em **DATA.** É uma boa idéia colocar sempre uma declaração **RESTORE** no início de um programa que você estiver desenvolvendo. Isso faz com que você não fique sem nenhuma **DATA,** ao tentar testá-lo. Se, mais tarde, você quiser remover a linha **RESTORE,** bastará colocar uma declaração **REM** após a linha em que ela está, para lembrá-lo.

Com o comando **RESTORE** você pode reutilizar uma lista **DATA** sempre que necessário. Ele é particularmente útil quando se quer ler várias vezes uma mesma lista de dados para pesquisar um determinado item, como no caso do programa para a lista telefônica.

APLICAÇÕES PARA DATA

As declarações **DATA** são úteis para todos os tipos de programa e você certamente já as utilizou em desenhos de labirintos e outros efeitos.

Freqüentemente, um grande número de linhas **DATA** é utilizado para conter todo o texto necessário a um jogo de aventuras, ou em programas aplicativos simples. Pode-se também armazenar questionários com todas as perguntas e respostas em linhas **DATA.** E os jogos de fliperama em BASIC as utilizam para definir tanto os caracteres como os planos de fundo.

Programadores experientes empregam **DATA** para linguagem Assembly ou em programas em código de máquina (veja a primeira lição de *Código de Máquina*). Tais programas consistem de um laço **FOR...NEXT** curto que lê os códigos de máquina em uma lista **DATA** e usa o comando **POKE** para inseri-los na memória.

Qualquer programa que contenha textos padronizados, figuras ou funções pode fazer um largo uso das listas **DATA.** Empregadas de maneira cuidadosa, essas listas constituem um poderoso instrumento de programação.

UTILIZE DATA PARA GRÁFICOS

As declarações **DATA** servem igualmente para programas de gráficos, para definir coordenadas, desenhar ou evocar caracteres gráficos predefinidos na memória ROM. Sua utilidade, porém, diminui quando o que se quer é traçar formas regulares onde as coordenadas ou o formato dos gráficos podem ser calculados. Um exemplo disso pode ser encontrado no programa para o TRS-Color, onde os padrões repetitivos no topo das muralhas de um forte são calculados.

As declarações **DATA** no programa abaixo foram deliberadamente divididas em várias linhas. Assim, se você quiser examinar ou modificar o programa futuramente e achar os itens em **DATA** que correspondem às variáveis do programa, poderá seguir uma a uma as instruções **READ**; mas para um programa longo isto será muito cansativo. O melhor será dividir as declarações **DATA** em grupos, e — se você quiser evitar a memorização de tudo o que está lá dentro — utilizar comandos **REM** para explicar para que serve cada grupo de declarações **DATA.**

O programa seguinte desenha uma casa:

```
10 CLS:COLOR 4,5
20 FOR A=0 TO 8
30 FOR B=1 TO 17
40 READ C
50 VPOKE (171+A*40)+B,C
60 NEXT B
70 NEXT A
500 DATA 32,32,32,32,32,32,32,3
2,32,32,32,32,219,219,219,32,32
510 DATA 32,32,32,32,32,32,32,3
2,32,32,32,219,219,219,219,219,
32
520 DATA 32,32,32,32,32,32,32,3
2,32,32,219,219,219,219,219,219
,219
530 DATA 32,32,32,199,219,219,3
2,32,32,32,32,222,219,219,219,2
19,219
540 DATA 32,32,199,219,219,219,
219,32,221,32,32,219,219,219,21
9,219,32
550 DATA 32,32,215,215,215,215,
215,219,219,193,32,32,222,215,2
19,32,32
560 DATA 32,32,215,16,215,202,2
15,215,21,215,32,32,32,215,32,3
2,32
570 DATA 32,32,215,215,215,202,
215,215,215,215,32,32,32,215,32
,32,32
580 DATA 195,195,195,195,195,19
5,195,195,195,195,195,195,195,1
95,195,195,195
```

Os números em **DATA** fornecem as coordenadas das várias partes da casa (linhas 500 a 580). A parte principal do programa estabelece os laços **FOR...NEXT** para a leitura (comando **READ** na linha 40) das partes pertinentes em **DATA** e as desenha, colocando os códigos de caracteres na página de memória da tela, por meio do comando **VPOKE.**

Este programa utiliza **READ...DATA** para desenhar uma ponte. Se você entrar o programa e rodá-lo em estágios, será bem mais fácil ver o que está acontecendo, e assim verificar se o digitou corretamente:

```
10 FOR t=74 TO 80 STEP 3
20 PLOT 35,t
30 DRAW 175,0,-2.5
40 NEXT t
```

Este segmento do programa utiliza um laço **FOR...NEXT** para ajudar a traçar três pontos próximos ao lado esquerdo da tela; em seguida, ele desenha uma linha em forma de arco a partir de cada ponto. A linha 30 significa: "desenhe até um ponto localizado 175 pixels à direita do pixel inicial". O parâmetro -2,5 dá à linha sua forma curva, evitando a linha reta.

```
100 FOR n=18 TO 39
110 READ a
120 PLOT n,45
130 DRAW 0,a
132 PLOT n+188,45
134 DRAW 0,a
140 NEXT n
1000 DATA 70,70,67,67,70,70,60,
60,57,57,60,60,57,57,60,60,70,7
0,67,67,70,70
```

Esta é a seção que desenha as torres. O laço entre as linhas 100 e 140, mais o número 45 (pixels da parte inferior da tela), nas linhas 120 e 132, traçam os pontos na parte inferior de cada um dos conjuntos de linhas verticais que definem a torre.

Então, as linhas 110 e 1000 assumem o comando. A linha 110 diz ao computador para ler, na linha 1000, a altura (novamente em pixels) de cada uma das 22 linhas verticais. Assim, a primeira linha será 0,70 — isto é, vertical e com 70 pixels de altura; a segunda linha será 0,70, a terceira 0,67 e assim por diante. Se você quiser ver o que está acontecendo, tente inserir uma linha temporária tal como **135 PAUSE 100** após a linha 134.

```
300 PLOT 0,75
310 DRAW 255,0-0.1
320 PLOT 0,78
330 DRAW 255,0,-0.1
```

Essas linhas desenham a pista de rolamento e -0.1 produz uma ligeira curva para cima.

```
400 FOR r=62 TO 182 STEP 20
410 PLOT r,78
420 READ b
430 DRAW 0,b
440 NEXT r
1010 DATA 42,55,63,65,63,55,42
```

Essas, por sua vez, produzem os cabos verticais entre o arco e a pista de rolamento; o laço **FOR...NEXT** ajuda a traçar as posições iniciais dos cabos, enquanto as linhas 420 e 1010 regulam as suas respectivas alturas.

Se você quiser que o programa rode de novo, automaticamente, acrescente essas linhas:

```
  5 CLS:RESTORE
450 GOTO 5
```

A linha 5 limpa a tela e permite que o programa leia novamente o **DATA.**

TRABALHE COM INDICADORES RESTORE

Até agora, o único problema com **DATA** se resumiu a que a informação precisou sempre ser chamada na mesma seqüência, começando do mesmo lugar. Utilizando o **RESTORE**, é possível voltar ao início de uma lista, mesmo que não se tenha atingido o fim. Mas como se salta para o meio de uma lista?

Nos computadores das linhas Sinclair Spectrum e MSX, existe um modo para se responder a essa questão. Em vez de uma lista apenas, você pode utilizar várias, direcionando assim o computador para a lista adequada. Tente acrescentar essas linhas extras ao programa para o Spectrum:

```
  6 INPUT "PONTE ANTIGA OU MOD
  ERNA?";   y$
  7 IF y$="a" THEN  RESTORE
  1000
  8 IF y$="m" THEN  RESTORE
  2000
  9 IF y$<>"a" AND y$<>"m"
  THEN  GOTO   5
 10 FOR t=74 TO 80 STEP 3
 20 PLOT 35,t
 30 DRAW 175,0,-2.5
 40 NEXT t
2000 DATA 83,84,85,86,87,88,89,
90,90,90,90,90,90,90,90,89,88,8
7,86,85,84,83
2010 DATA 42,55,63,65,63,55,42
```

Os números que seguem o comando **RESTORE** são conhecidos como indicadores de **RESTORE**. Eles direcionam o computador para uma lista **DATA**. No programa acima, se você pressionar a tecla O, o indicador **RESTORE** será definido com o 1000, na linha 7.

Se você quiser que o micro vá para a primeira lista **DATA** disponível no programa, não será necessário um indicador: usar o **RESTORE** sem um número de linha é o mesmo que restaurar o número da primeira linha **DATA**.

Indicadores de restauração são muito úteis em programação de jogos. Numa aterrissagem, por exemplo, você poderia escrever uma rotina para checar se houve uma colisão ou um pouso seguro. As listas **DATA** poderiam então conter informações para gerar sons para as duas alternativas, e o indicador **RESTORE** serviria para indicar a lista correta.

Outra vantagem do **RESTORE** indexado para definir os elementos de uma figura é que fica bem mais fácil acrescentar novos elementos a um desenho terminado; se quisermos adicionar um muro à casinha da ilustração abaixo, bastará colocar mais linhas **DATA**:

```
80 LOCATE 0,21:PRINT"Pressione
qualquer tecla para uma casa co
m muro!";
90 IFINKEY$<>""THENRESTORE520:G
OTO10:ELSEGOTO90
590 DATA 203,203,203,203,203,20
3,203,203,203,203,203,203,203,2
03,203,203,203
600 DATA 203,203,203,203,203,20
3,203,203,203,203,203,203,203,2
03,203,203,203
```

Feito o desenho, o computador perguntará se você quer uma casa com muro. Se você pressionar qualquer tecla, o comando **RESTORE 520** na linha 90 indicará, para a próxima execução do programa (a partir da linha 10), que as linhas **DATA** lidas pelos comandos **READ** na linha 40 deverão começar na linha 520, e não em 500, que é o seu início físico. De acordo com uma nova seqüência de caracteres gráficos (que termina com as linhas adicionais 590 e 600), o desenho será diferente do anterior.

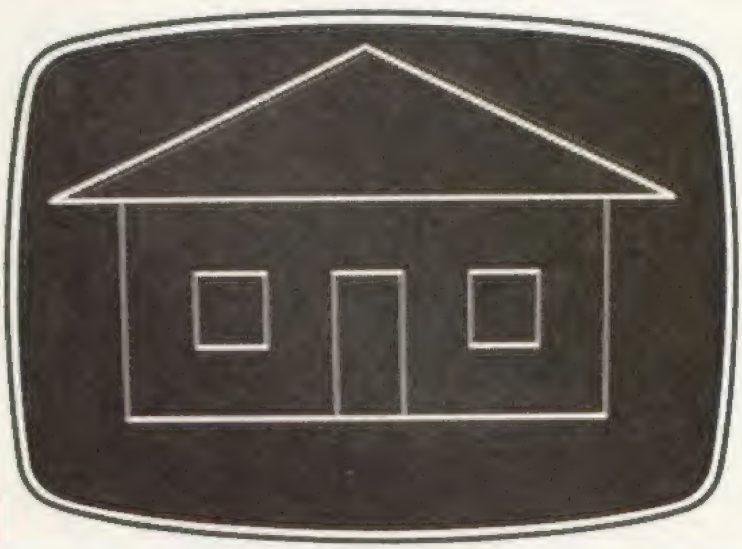
O programa a seguir construirá um castelo:

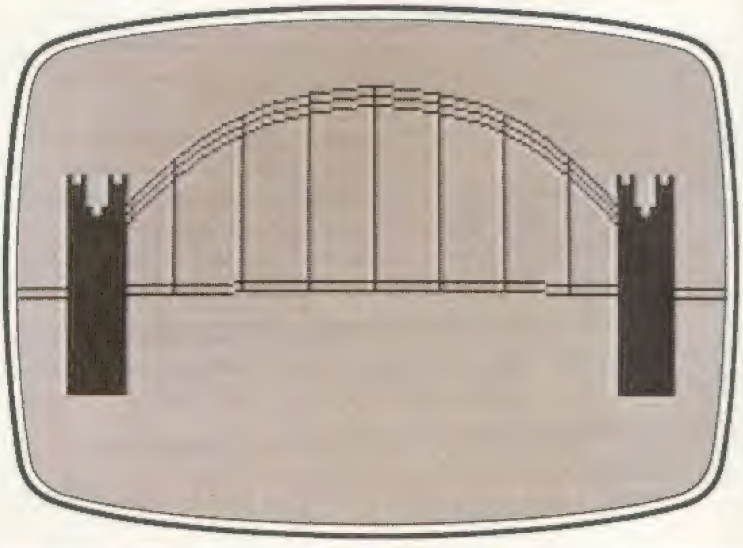
```
10 PCLEAR 4
20 PMODE 4,1
30 PCLS 5
40 SCREEN 1,1
50 READ SX,SY
60 LINE -(SX,SY),PSET
70 FOR K=1 TO 18
80 READ X,Y
90 LINE -(X,Y),PRESET
100 NEXT K
270 GOTO 270
500 DATA 64,160
510 DATA 64,60,32,60,48,40,64,6
0,32,60,32,160,110,160,110,120
520 DATA 152,120,152,160,228,16
0,228,60,212,40,196,60,228,60
530 DATA 196,60,196,160,196,74
```

As linhas de 10 a 40 deixam a tela para gráficos de alta resolução pronta para desenhar. A linha 270 a mantém ligada. As linhas 50 e 60 são um tanto incomuns. As coordenadas do ponto inicial do castelo são lidas, e uma linha em branco é desenhada em fundo branco para o ponto inicial. Algumas vezes, essa capacidade para desenhar linhas "invisíveis" torna-se bastante útil, pois elas podem se juntar às linhas visíveis em um esquema contínuo de programação.

Da linha 70 à 100, é desenhado o contorno do castelo. No final desta parte do programa foram lidos 38 itens de **DATA**. Agora acrescente essas linhas e construa algumas muralhas:

```
110 FOR K=1 TO 33
120 LET X=X-4
130 LINE-(X,Y),PRESET
140 IF Y=74 THEN LET Y=78 ELSE
```





```
LET Y=74
150 LINE -(X,Y),PRESET
160 NEXT K
```

Provavelmente você está tentando adivinhar o que aconteceu à linha **READ**. Esta é uma ocasião em que as instruções **DATA** e **READ** não representam economia de trabalho, pois você precisaria de 66 itens de **DATA** para definir os cantos da muralha. Programando desta forma, você não precisará digitar todas as linhas **DATA** extras.

Essas linhas desenharão as janelas:

```
170 FOR K=1 TO 8
180 READ X,Y
190 LINE (X,Y) - (X+4,Y),PRESET
200 LINE (X+2,Y-2)-(X+2,Y+6),PR
ESET
210 NEXT K
540 DATA 46,80,46,120,210,80,21
0,120,86,90,170,90,80,132,178,1
32
```

Você não precisa do **DATA** para finalizar as janelas. Como elas são todas do mesmo tamanho, apenas um conjunto de **DATA** é necessário. Acrescente agora essas linhas, rode o programa e veja o que acontece:

```
220 FOR K=1 TO 4000
230 NEXT K
240 CLS: PRINT @33,"PRESSIONE Q
UALQUER TECLA PARA   CONTINUAR"
250 LET IN$=INKEY$:IF IN$="" TH
EN GOTO 250
270 GOTO 30
```

Quando você pressionar uma tecla qualquer, obterá OD ou a mensagem de erro "out of data". A razão disso é que o programa chegou ao final da lista **DATA** — não existe mais nenhuma depois dele. Mas você não precisa digitar todos os **DATA** novamente. Acrescente só esta linha, que utiliza a declaração **RESTORE** para instruir o computador a retornar ao início da lista **DATA:**

```
260 RESTORE
```

Desenhar uma casinha nos micros da linha Apple é facílimo quando utilizamos **DATA**. Digite o programa abaixo:

```
10 HGR
20 HCOLOR=3
30 FOR I=1 TO 4
40 READ XC,YC,LX,LY
50 HPLOT XC,YC TO XC+LX,YC TO
   XC+LX,YC+LY TO XC,YC+LY TO
   XC,YC
60 NEXT I
80 FOR I=1 TO 3
90 READ XO,YO,XD,YD
100 HPLOT XO,YO TO XD,YD
110 NEXT I
400 DATA 20,50,160,80
410 DATA 90,80,20,50
420 DATA 40,80,30,30
430 DATA 130,80,30,30
440 DATA 0,50,200,50
450 DATA 200,50,100,0
460 DATA 100,0,0,50
```

Os elementos em **DATA** fornecem as coordenadas para as diferentes partes da casa.

A parte principal do programa liga o modo de alta resolução gráfica (HGR) e lê os dados contidos em **DATA** por intermédio do laço **FOR...NEXT** nas linhas 30 a 60 e 80 a 110.

Os dados contidos nas linhas **DATA** que vão de 400 a 430 contêm quatro números cada e servem para traçar retângulos de qualquer tamanho e em qualquer posição na tela (rotina **HPLOT** na linha 50). Essa estratégia é vantajosa, pois você pode notar na ilustração que a casa tem no mínimo quatro retângulos diferentes (a casa, a porta e as duas janelas). Assim, os dois primeiros números em **DATA** são as coordenadas XC e YC do ponto de origem do retângulo na tela (seu canto superior esquerdo). Os dois números seguintes são: o comprimento do lado paralelo ao eixo horizontal (LX) e o do lado paralelo ao eixo vertical (LY). Assim, o laço **FOR...NEXT** das linhas 30 e 60 lê sucessivamente quatro conjuntos de dados e traça os quatro retângulos.

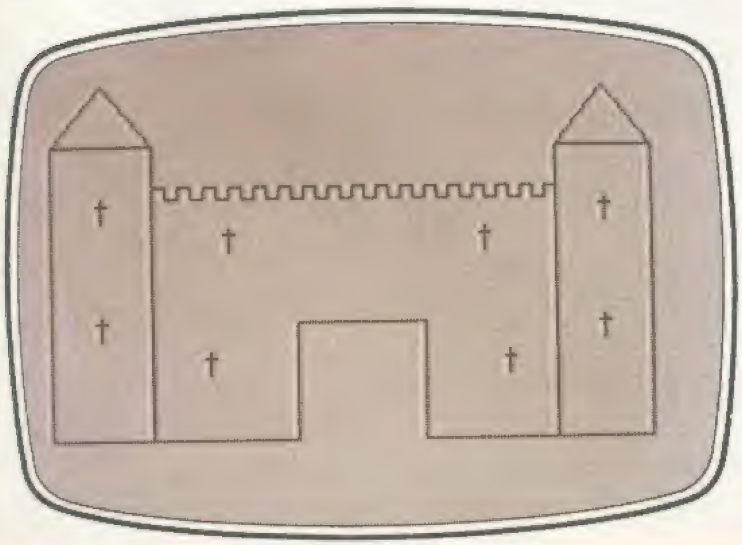
Já o traçado do triângulo exige uma técnica diferente. O laço que vai das linhas 80 a 110 lê 4 valores: XO, YO (coordenadas de origem de uma reta) e XD, YD (coordenadas do seu ponto final), e traça essa reta, na linha 90.

Uma vez que você tenha rodado o programa, talvez queira repeti-lo mais uma vez. Para isso, tente acrescentar as linhas seguintes, rode o programa e veja o que acontece:

```
120 HOME
130 PRINT "QUER VER DE NOVO
   (S/N)?":GET R$
140 IF R$="N" THEN TEXT:END
150 GOTO 10
```

Como vimos antes, se você pressionar uma tecla qualquer obterá OD ou a mensagem de erro "out of data". A razão disso é que o programa chegou ao final da lista **DATA**. Felizmente, você não precisa digitar todos os **DATA** novamente. Apenas acrescente a linha seguinte, que utiliza a declaração **RESTORE** para instruir o computador a retornar ao início da lista **DATA:**

```
150 RESTORE:GOTO 10
```



A casa da página anterior foi desenhada por um Apple II a partir de quatro linhas **DATA**. Para fazer a ponte, o Spectrum utilizou **PLOT** e **DRAW**, junto com **DATA**. Ao lado, castelo construído em etapas pelos TRS-Color.

MICRO DICAS

ORGANIZE MELHOR OS COMANDOS DATA EM UM PROGRAMA

As linhas **DATA** têm quase sempre uma coisa em comum: qualquer que seja a informação que elas transportam, leva-se muito tempo para digitá-las e fazê-las funcionar. Tudo correrá bem na primeira vez, mas pode ser que você precise retornar posteriormente a um programa e não consiga entender mais nada do que foi colocado lá.

Organizar sistematicamente as linhas **DATA** do programa toma apenas um pouco mais de tempo, mas pode poupar séculos no resultado final.

Onde o **DATA** definir um gráfico por blocos ou algo parecido, organize as linhas do programa para corresponder diretamente às linhas do gráfico. Se o **DATA** segue um formato repetitivo (tal como os dados em uma lista telefônica, por exemplo), classifique cada entrada exatamente do mesmo modo nas linhas do programa.

PONHA ORDEM EM SUAS CONTAS

Assim como os grandes computadores comerciais, seu micro pode ser usado para fazer cálculos e manter registros financeiros. Veja como fazer isso e facilite sua vida.

Neste artigo, examinaremos um programa de contabilidade doméstica, que foi planejado para fornecer respostas a perguntas como: onde foi parar o dinheiro do salário? São apresentadas versões para todos os computadores, com exceção dos compatíveis com o ZX-81.

Para atualizar a sua contabilidade, você deve "alimentá-la" uma vez por mês com seus rendimentos e despesas.

O programa cria uma espécie de diário contendo uma coluna para os rendimentos e sete colunas para as despesas sob diferentes rubricas. As subdivisões referentes às despesas podem ser modificadas, caso necessário; para isso, altere os dizeres (identificação das categorias de despesa) na declaração **DATA** pertinente, quando digitá-las, junto com o programa. Mas existem algumas limitações: a coluna de rendimentos deve aparecer sempre em último lugar; ao mesmo tempo, é necessário que haja um total de oito colunas.

O programa deve ser digitado em duas seções: a primeira, com o próprio programa; e a segunda, com todas as informações com as quais você o alimentou até o momento da sua entrada. Isto significa que você precisará de dois nomes para os programas.

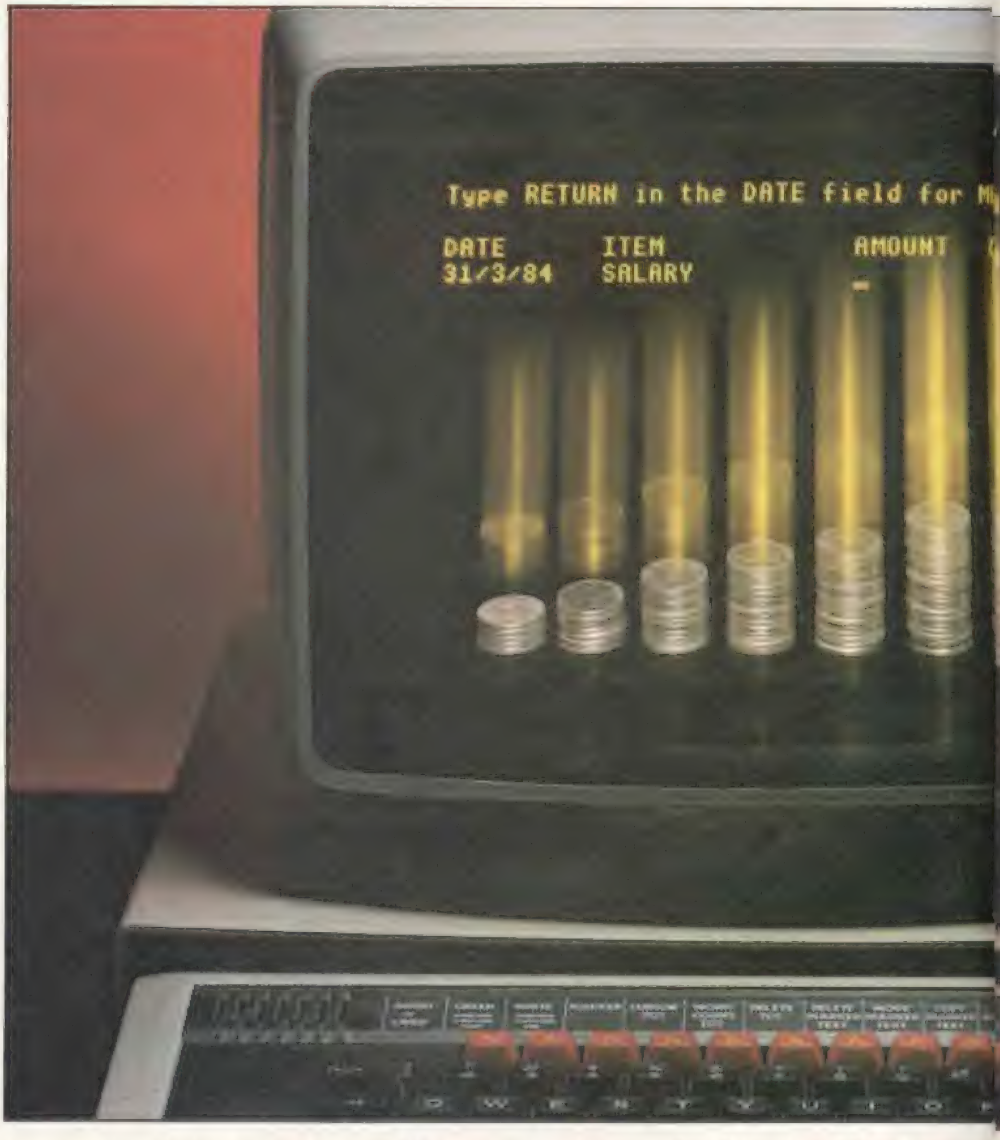
Para gravar corretamente o programa em fita, siga o procedimento normal de sua máquina (comando **SAVE** ou **CSAVE**). Para recarregar o programa, siga novamente o procedimento usual para carregar jogos gravados em fitas.

Quando você rodar o programa, o menu principal lhe dará sete opções:

1 - Entrar dados
2 - Ver dados
3 - Gravar em fita
4 - Carregar da fita
5 - Imprimir
6 - Alterar um dado
7 - Encerrar o programa

Para fazer um lançamento, pressione a tecla 1 quando o menu principal aparecer. Não acione ainda <**ENTER**> ou <**RETURN**>. A cada lançamento, o computador pedirá detalhes como data, descrição, valor e categoria.

Digite a informação na ordem dada, pressionando <**ENTER**> ou <**RETURN**> após cada dado do lançamento. Espere que o computador pergunte por um novo dado e pressione <**ENTER**> ou <**RETURN**>, trazendo de volta o menu.



Para examinar um lançamento, pressione a tecla 2 quando o menu principal aparecer. Não tecle <**ENTER**> ou <**RETURN**>. O computador exibirá um índice mostrando as várias categorias. Para selecionar uma delas digite o número apropriado (não tecle <**ENTER**> ou <**RETURN**>, e o micro listará os lançamentos que existem para aquela categoria.

Na versão para o Spectrum, a tela mostrará a questão "Continua?" se houver espaço insuficiente para exibir todos os lançamentos de uma vez. Neste caso, não pressione o N nesse ponto: prossiga até o final da listagem.

Quando terminar, acione <**ENTER**> ou <**RETURN**> e volte ao menu principal.

COMO MODIFICAR UM LANÇAMENTO

Quando você teclar o 6 para a opção de alterar um lançamento, o computa-

- DIGITE E ARMAZENE O PROGRAMA
- AS OPÇÕES DO MENU
- COMO ENTRAR LANÇAMENTOS NOS REGISTROS
- ATUALIZE OS REGISTROS
- VERIFICAÇÃO DO BALANÇO
- IMPRESSÃO DOS RESULTADOS
- ARMAZENE SUAS FINANÇAS EM FITA

dor mostrará uma lista com todos os lançamentos já efetuados.

Você pode se movimentar para trás e para a frente dentro da lista, utilizando os sinais de prontidão que aparecerão na tela. O computador também dirá como editar (modificar) um lançamento. Uma vez que você tenha digitado <**ENTER** > ou <**RETURN**>, após fazer a alteração, o computador trará de volta o menu principal. Para uma segunda alteração, você deve selecionar a opção 6 novamente. O comando para a opção de impressão funciona de forma tão simples quanto um interruptor de luz — um liga/desliga acionado quando se tecla a opção 5 (também sem <**ENTER**> ou <**RETURN**>) do menu principal. O computador pedirá que você tecle S, se quiser usar a impressora, ou N se não quiser. Se você pressionar a tecla S, o menu principal voltará automaticamente à tela e, a partir desse momento, tudo o que normalmente apareceria na tela, usando-se a opção 2, passará a ser mandado para a impressora. Se quiser interromper essa opção, pressione a tecla 5 novamente e responda N.

Se você não tiver uma impressora conectada ao computador, tome cuidado para não pressionar a tecla S. O Spectrum irá ignorar as instruções neste caso, mas em qualquer um dos outros três computadores você poderá perder todas as informações já inseridas.

```
10 PMODE 0:PCLEAR 1:CLEAR 10000
20 DIM TE$(200),AM(200),DA$(200
),CT$(8),CA(200)
30 FOR N=1 TO 8:READ CT$(N):NEX
T
40 DATA MANUTENCAO DA CASA,LAZE
R ,ALUGUEL,VESTUARIO,AUTOMOVEL,
FERIAS,OUTROS,RENDA
50 U1$="#########,":U2$="######
##."
60 CLS4: PRINT @8,"MENU PRINCIP
AL";:PRINT @70,"1- ENTRAR DADOS
    ";:PRINT @134,"2- VER ENTRA
DAS    ";:PRINT @198,"3- GRAVAR
 EM FITA  ";
70 PRINT @262,"4- CARREGAR
   ";:PRINT @326,"5- IMPRIMIR
     ";:PRINT @390,"6- MUDAR E
NTRADA    ";:PRINT @454,"7- SAID
A          ";
80 A$=INKEY$:IF A$<"1" OR A$>"7
" THEN 80
90 ON VAL(A$) GOSUB 1000,2000,3
000,4000,5000,6000,7000
100 GOTO 60
1000 CLS: IF NU>200 THEN PRINT
@264,"MEMORIA LOTADA !":PLAY "T
10ABCDEFGP1P1":RETURN
1010 GOSUB 1160
1020 GOSUB 1250:INPUT"DATA ";DA
$(NU)
1030 IF DA$(NU)="" THEN RETURN
1040 IF LEN(DA$(NU))>8 THEN 102
0
1050 PRINT @L,DA$(NU);
1060 GOSUB 1250:LINEINPUT"ITEM
?";TE$(NU):IF LEN(TE$(NU))>25 T
HEN 1060
1070 A$=LEFT$(TE$(NU),9):PRINT@
L+14-LEN(A$)/2,A$;
1080 GOSUB 1250:INPUT"QUANTIA "
;A
1090 IF A>9999999 OR A<0 THEN 1
080
1100 PRINT @L+19,USING U2$;A;:A
M(NU)=A
1110 GOSUB 1250:INPUT"CATEGORIA
 ";CA$
1120 GOSUB 1180:IF F=0 THEN 111
0
1130 CA(NU)=NM:PRINT @L+29, LEF
T$(CT$(CA(NU)),3);
1140 IF NM<>8 THEN GT=GT+A
1150 NU=NU+1: L=L+32:IF L=448 T
HEN 1000 ELSE 1020
1160 L=64: PRINT @2,"DATA" TAB(
10)"ITEM" TAB(19) "QUANTIA" TAB
(29)"CAT";
1170 RETURN
1180 IF VAL(CA$)<>0 THEN 1230
1190 F=0:FOR N=1 TO 8
1200 IF CA$=LEFT$(CT$(N),LEN(CA
$)) THEN F=F+1:NM=N
1210 NEXT: IF F>1 THEN F=0
1220 RETURN
1230 IF VAL(CA$)>8 THEN F=0:RET
URN
1240 NM=VAL(CA$):F=1:RETURN
1250 PRINT @448," ":PRINT @449,
;:RETURN
2000 CLS3:FOR N=1 TO 8
2010 PRINT @69+N*32,N;MID$("- "
+CT$(N)+STRING$(12," "),1,20);
2020 NEXT
2030 TT=0: PRINT @449,"QUE CATE
GORIA ?";
2040 A$=INKEY$:IF A$<"1" OR A$>
"8" THEN 2040
2050 NM=VAL(A$)
2060 IF PT=1 THEN PRINT#-2,CHR$
(13):PRINT#-2,TAB(21-LEN(CT$(NM
))/2);CT$(NM):PRINT#-2,"  DATA"
TAB(20)"ITEM"TAB(39)"QUANTIA"
2070 GOSUB 2280
2080 FOR NN=0 TO NU
2090 IF CA(NN)<>NM THEN 2150
2100 IF PT=1 THEN PRINT#-2,USIN
G F$;DA$(NN);TE$(NN);AM(NN)
2110 PRINT @L,DA$(NN);:A$=LEFT$
(TE$(NN),13):PRINT @L+16-LEN(A$
)/2,A$;:PRINT @L+23,USING U2$;A
M(NN);:TT=TT+AM(NN)
2120 L=L+32:IF (L=448 AND NM<>8
) OR (L=352 AND NM=8) THEN PRIN
T @465,"CONT ?";ELSE GOTO 2150
2130 A$=INKEY$:IF A$="" THEN 21
30
2140 GOSUB 2280
```

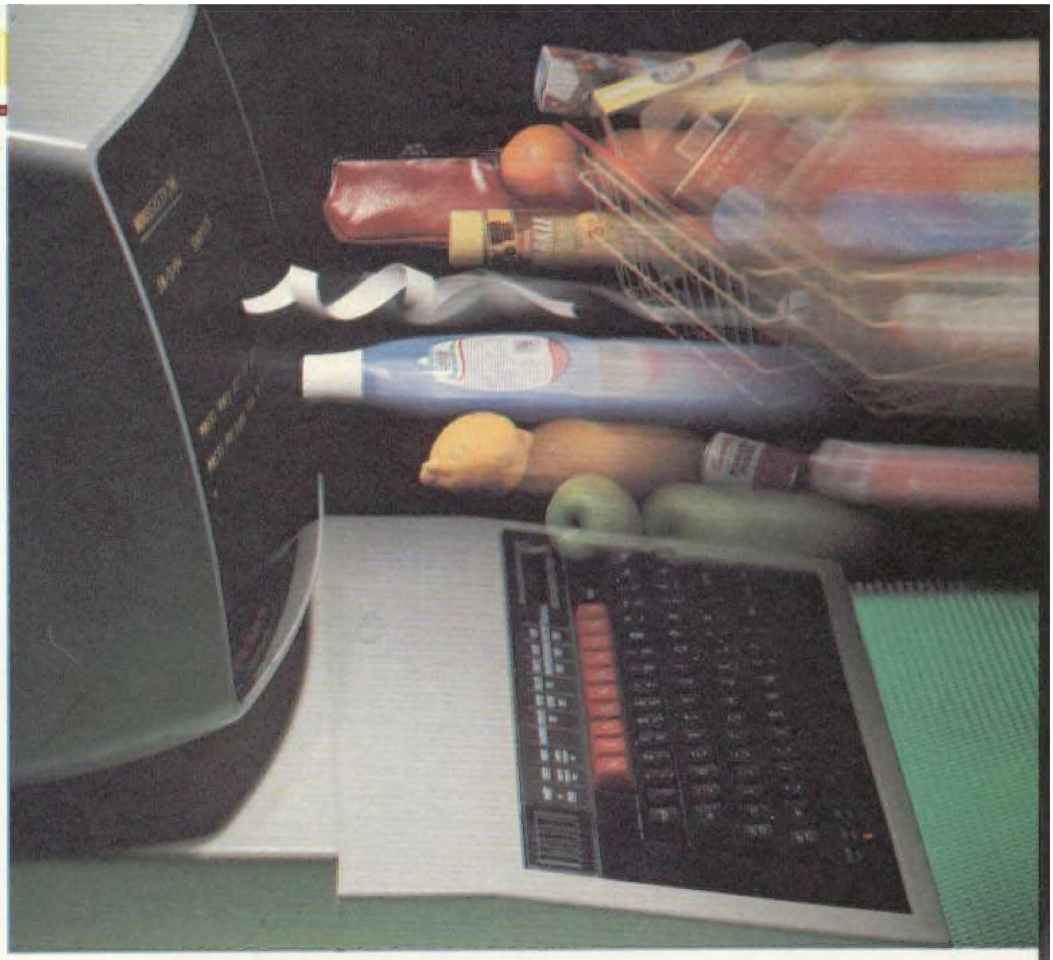
```
2150 NEXT: IF PT=1 THEN PRINT#-
2,CHR$(13): IF NM<>8 THEN PRINT
#-2,TAB(28);:PRINT#-2,USING"TOT
AL ="+U1$;TT
2160 PRINT @463,USING "TOTAL ="
+U1$;TT;
2170 IF NM<>8 THEN 2250
2180 IF PT=0 THEN 2220
2190 PRINT#-2,TAB(21);:PRINT#-2
,USING"RENDA TOTAL ="+U1$;TT
2200 PRINT #-2,TAB(16);:PRINT#-
2,USING"DESPESA TOTAL ="+U1$;GT
:PRINT#-2,TAB(35)"              "
2210 PRINT#-2,TAB(26);:PRINT#-2
,USING"BALANCO ="+U1$;TT;
2220 PRINT @392,USING "RENDA TO
TAL ="+U1$;TT;
2230 PRINT @419,USING "DESPESA
TOTAL ="+U1$;GT;
2240 PRINT @461,USING"BALANCO =
"+U1$;TT-GT ;
2250 A$=INKEY$:IF A$="" THEN 22
50
2260 IF A$<>CHR$(13) THEN 2000
2270 RETURN
2280 L=64:CLS NM:PRINT @(33-LEN
(CT$(NM)))/2,CT$(NM);
2290 PRINT @34,"DATA";:PRINT @4
5,"ITEM";:PRINT @56,"QUANTIA";
2300 FOR N=32 TO 416 STEP 32
2310 POKE N+1032,122+NM*16:POKE
 N+1046,117+16*NM
2320 NEXT:RETURN
3000 CLS:MOTORON:PRINT @64,"POS
ICIONE O GRAVADOR E PRESSIONE<E
NTER>"
3010 A$=INKEY$:IF A$="" THEN 30
10
3020 MOTOROFF:PRINT @65,"PRESSI
ONE 'REC' NO GRAVADOR      E TEC
LE <ENTER>"
3030 A$=INKEY$:IF A$="" THEN 30
30
3040 CLS: PRINT @65,;:INPUT"NOM
E DO ARQUIVO ";DA$
3050 OPEN "O",#-1,DA$
3060 PRINT #-1,NU
3070 FOR N=0 TO NU-1
3080 PRINT #-1,DA$(N),TE$(N),AM
(N),CA(N)
3090 NEXT: CLOSE#-1:RETURN
4000 CLS: PRINT @65,;:INPUT"NOM
E DO ARQUIVO";DA$
4010 MOTORON:PRINT @64,"POSICIO
NE O GRAVADOR E TECLE     <ENTER
>"
4020 A$=INKEY$:IF A$="" THEN 40
20
4030 MOTOROFF:GT=0:OPEN "I",#-1
,DA$
4040 PRINT @129,"ACHEI ";DA$
4050 INPUT #-1,NU
4060 FOR N=0 TO NU-1
4070 INPUT #-1,DA$(N),TE$(N),AM
(N),CA(N)
4080 IF CA(N)<>8 THEN GT=GT+AM(
N)
4090 NEXT: CLOSE #-1: RETURN
5000 CLS: PRINT @65, "VOCE QUER
 IMPRIMIR ? (S/N)";
5010 A$=INKEY$:IF A$<>"S" AND A
$<>"N" THEN 5010
5020 PRINT "OK":IF A$="N" THEN
PT=0:RETURN
5030 F$="%         %%
            %    "+U2$:PT=1:RETU
RN
6000 IF NU=0 THEN RETURN
6010 CLS: PRINT"  DATA"TAB(11)"
ITEM"TAB(19)"QUANTIA"TAB(29)"CA
T"
6020 PRINT @417,"PRESSIONE <";C
HR$(94);"> PARA PROSSEGUIR
OU <";CHR$(95);"> PARA RETORNAR
 OU"
6030 PRINT @481,"A BARRA DE ESP
ACOS PARA EDITAR";:M=0:GOTO 608
0
6035 KK$=INKEY$:IFKK$=""THEN603
5
6040 IF KK$=CHR$(8) AND M>0 THE
N M=M-1:GOTO 6080
6050 IF KK$=CHR$(94) AND M<NU-1
 THEN M=M+1:GOTO 6080
6060 IF KK$=CHR$(32) THEN 6100
6070 GOTO 6035
6080 PRINT@64,USING"%       % %
      %########, %  %";DA$(M);
LEFT$(TE$(M),8);AM(M);LEFT$(CT$
(CA(M)),3)
6090 GOTO 6035
6100 IF CA(M)<>8 THEN GT=GT-AM(
M)
6110 INPUT"NOVA DATA ";D$:IF D$
="" THEN 6130
6120 IF LEN(D$)>8 THEN 6110 ELS
E DA$(M)=D$
6130 INPUT "NOVO ITEM ";D$:IF D
$="" THEN 6150
6140 TE$(M)=D$
6150 INPUT "NOVA QUANTIA ";A:IF
 A=0 THEN 6170
6160 IF A<0 OR A>9999999 THEN 6
150 ELSE AM(M)=A
6170 INPUT "NOVA CATEGORIA ";CA
$:IF CA$="" THEN 6200
6180 GOSUB 1180:IF F=0 THEN 617
0
6190 CA(M)=NM
6200 IF CA(M)<>8 THEN GT=GT+AM(
M)
6210 RETURN
7000 CLS: PRINT @69,"VOCE TEM C
ERTEZA (S/N) ?;
7010 A$=INKEY$:IF A$<>"S" AND A
$<>"N" THEN 7010
7020 IF A$="N" THEN RETURN
```

```
 50 LET mn=200: IF PEEK 23733=
 127 THEN  LET mn=100
100 DIM c$(8,16): DIM a(mn):
DIM a$(mn,23)
110 LET u=0: LET v=1
120 FOR n=v TO 8: READ c$(n):
NEXT n
130 POKE 23658,8
140 LET k$=".00": FOR n=v TO 7
145 LET k$=k$+CHR$ 8: NEXT n
190 LET p=2: LET tt=u: LET cr=
u
200 CLS : PRINT  BRIGHT v;
PAPER 2; INK 6;AT 2,1;" M E N
U   P R I N C I P A L "
210 PRINT  BRIGHT v; PAPER 7;
AT 5,4;" 1- ENTRAR REGISTRO ";
AT 7,4;" 2- VER REGISTROS   ";
AT 9,4;" 3- GRAVAR EM FITA  ";
AT 11,4;" 4- CARREGAR       "
;AT 13,4;" 5- IMPRIMIR
";AT 15,4;" 6- MUDAR REGISTRO
 ";AT 17,4;" 7- SAIDA
 "
220 PRINT  INK 3; FLASH v;
BRIGHT v;AT 20,4;" - SELECIONE
 OPCAO - "
```

```
 230 IF INKEY$="" THEN  GOTO
  230
 240 LET z$=INKEY$: IF z$<"1"
 OR z$>"7" THEN  GOTO 230
 250 CLS : GOSUB 1000*VAL z$
 260 GOTO 200
1000 LET c=u
1005 LET c=c+v: IF c=mn+v THEN
 RETURN
1006 IF a$(c,v)=" " THEN  GOTO
1010
1007 GOTO 1005
1010 PRINT AT u,u; BRIGHT v; PA
PER 2; INK 7;" DATA      ITEM
   QUANTIA CAT "
1015 IF c=mn+v THEN  RETURN
1020 INPUT "Digite a data"; LIN
E a$(c,2 TO 9): IF a$(c,2)=" "
THEN  RETURN
1030 PRINT TAB u;a$(c,2 TO 9);
1040 INPUT "Digite o item "; LI
NE a$(c,10 TO 23): IF a$(c,10)=
" " THEN  GOTO 1040
1050 PRINT TAB 9;a$(c,10 TO 21)
;
1060 INPUT "Quantia ";a(c): IF
a(c)=u THEN  GOTO 1060
1070 LET vv=a(c)*100: LET v$=ST
R$ vv: PRINT TAB 27-LEN v$;a(c)
;
1080 INPUT "Categoria "; LINE f
$: IF f$="" THEN  GOTO 1080
1090 FOR n=v TO 8: IF f$=c$(n,v
 TO LEN f$) THEN  GOTO 1130
1100 NEXT n: GOTO 1080
1130 IF n=8 THEN  LET cr=cr+a(c
)
1140 IF n<>8 THEN  LET tt=tt+a(
c)
1150 PRINT TAB 29;c$(n,v TO 3)
1160 LET a$(c,v)=CHR$ (48+n)
1200 LET c=c+v: GOTO 1015
2000 FOR n=v TO 8: PRINT  PAPER
 v; INK 7;AT n*2,4;" ";n;"- ";c
$(n): NEXT n
2010 PRINT  FLASH v; INK 2;AT 1
9,3;" Selecione categoria(1 a 8
) "
2020 IF INKEY$="" THEN  GOTO 20
20
2030 LET z$=INKEY$: IF z$<"1" O
R z$>"8" THEN  GOTO 2020
2040 LET t=u: LET c=u
2050 CLS : PRINT #p; PAPER 6; B
RIGHT v;TAB 10;c$(VAL z$);TAB 3
1;" "
2055 LET c=c+v: IF c=mn THEN  G
OTO 2500
2060 IF a$(c,v)=" " THEN  GOTO
2500
2070 IF a$(c,v)<>z$ THEN  GOTO
2055
2080 PRINT #p;a$(c,2 TO 9);TAB
10;a$(c,10 TO 23);
2090 LET am=a(c)*100: LET n$=ST
R$ am: PRINT #p;TAB 29;k$;TAB 3
1-LEN n$;a(c)
2100 LET t=t+a(c)
2110 GOTO 2055
2500 PRINT #p;TAB 25;"-------":
 LET tx=t*100: LET n$=STR$ tx
2505 PRINT #p;TAB 12;"TOTAL- ";
TAB 29;k$;TAB 31-LEN n$;t
2510 IF z$<>"8" THEN  GOTO 2590
2520 LET tz=tt*100: LET n$=STR$
 tz: PRINT '#p;TAB 4;"DESPESA T
OTAL- ";TAB 29;k$;TAB 31-LEN n$
;tt
2530 LET ba=(t-tt)*100: LET n$=
STR$ ba: PRINT '#p;TAB 10;"BALA
NCO- ";TAB 29;k$;TAB 31-LEN n$;
ba/100
2590 PRINT  PAPER 2; INK 7'"Pre
ssione qualquer tecla para  co
ntinuar"
2600 PAUSE u: IF PEEK 23560=13
THEN  RETURN
2610 CLS : GOTO 2000
3000 GOSUB 8000: IF re=v THEN
RETURN
3010 PRINT  PAPER 6;AT 10,u;" D
igite o nome do arquivo data "
3015 INPUT  LINE w$: IF LEN w$>
10 OR LEN w$<v THEN  GOTO 3010
3020 CLS : SAVE w$ DATA a(): SA
VE w$ DATA a$(): RETURN
4000 GOSUB 8000: IF re=v THEN
RETURN
4010 PRINT  BRIGHT v;AT 10,u;"D
igite o nome dos dados a serem
carregados": INPUT  LINE w$. IF
 LEN w$>10 THEN  GOTO 4010
4020 PRINT  PAPER 3; INK 7;AT 1
0,u;"              Pressione  PLA
Y                  "
4030 LOAD w$ DATA a(): LOAD w$
DATA a$()
4040 LET cr=u: LET tt=u: FOR n=
v TO mn: IF a$(n,v)="8" THEN  L
ET cr=cr+a(n)
4050 IF a$(n,1)<>"8" THEN  LET
tt=tt+a(n)
4060 NEXT n: RETURN
5000 PRINT  BRIGHT v;AT 10,u;"
Voce quer imprimir (S/N)? "
5010 PAUSE u: IF INKEY$="" THEN
  GOTO 5010
5020 LET z$=INKEY$
5030 IF z$="n" THEN  LET p=2: R
ETURN
5040 IF z$="s" THEN  LET p=3: R
ETURN
5050 GOTO 5010
6000 LET c=v: IF a(c)=u THEN  R
ETURN
6010 PRINT AT u,u; BRIGHT v; PA
PER (VAL a$(c,v))-v; INK 9;" Nu
mero ";c,c$(VAL a$(c,v))
6015 PRINT  PAPER 2; INK 7;''"D
ATA      ITEM          QUANTIA
": PRINT 'a$(c,2 TO 9);TAB 10;a
$(c,10 TO 23);
6020 LET am=a(c)*100: LET n$=ST
R$ am: PRINT TAB 29;k$;TAB 31-L
EN n$;a(c)
6030 PRINT  PAPER 3; INK 7;AT 2
0,u;" A - Prossegue   Q - Retor
na          EDIT para alterar re
gistro "
6040 PAUSE u
6050 IF INKEY$="q" AND c>v THEN
  LET c=c-v: GOTO 6010
6060 IF INKEY$="a" AND c<>mn TH
EN  LET c=c+v
6070 IF a(c)=u THEN  LET c=c-v
6080 IF PEEK 23560=7 THEN  GOTO
 6100
6090 GOTO 6010
6100 INPUT  BRIGHT v;"Digite a
nova data "; LINE a$(c,2 TO 9):
 IF a$(c,2)=" " THEN  GOTO 6100
6110 PRINT AT 5,u;a$(c,2 TO 9)
6120 INPUT  BRIGHT v;"Digite o
novo item "; LINE a$(c,10 TO 23
): IF a$(c,10)=" " THEN  GOTO 6
120
6130 PRINT AT 5,10;a$(c,10 TO 2
3)
6135 IF a$(c,v)="8" THEN  LET c
r=cr-a(c)
6136 IF a$(c,v)<>"8" THEN  LET
```

```
tt=tt-a(c)
6140 INPUT  BRIGHT v;"Digite no
va quantia ";a(c): IF a(c)=u TH
EN  GOTO 6140
6150 LET am=a(c)*100: LET n$=ST
R$ am: PRINT AT 5,29;k$;TAB 31-
LEN n$;a(c)
6160 INPUT  BRIGHT v;"Digite a
nova categoria "; LINE f$: IF f
$="" THEN  GOTO 6160
6170 FOR n=v TO 8: IF f$=c$(n,v
 TO LEN f$) THEN  GOTO 6190
6180 NEXT n: GOTO 6160
6190 LET a$(c,v)=CHR$ (48+n)
6200 IF n=8 THEN  LET cr=cr+a(c
)
6210 IF n<8 THEN  LET tt=tt+a(c
)
6220 RETURN
7000 GOSUB 8000: IF re=v THEN
RETURN
7010 RAND USR u
8000 PRINT  PAPER 4;AT 10,9;" V
oce tem certeza? "
8010 PAUSE u: LET re=u: IF INKE
Y$<>"s" THEN  LET re=v
8020 RETURN
9000 DATA "MANUTENCAO CASA","LA
ZER","ALUGUEL E TAXAS","VESTUAR
IO","AUTOMOVEL","FERIAS","OUTRO
S","RENDA"
```

```
10 COLOR 15,4,4:KEYOFF:MOTOROFF
:CLEAR5000
20 DIMTE$(200),AM(200),DA$(200)
,CT$(8),CA(200)
30 FORN=1TO8:READCT$(N):NEXT
40 DATA MANUTENÇÃO DA CASA,LAZE
R,ALUGUéIS E TAXAS,ROUPAS,AUTOM
óVEL,FERIAS,MISCELÂNEA,RENDAS
50 U1$="$$##########.":U2$="###
####.##"
60 CLS:LOCATE7,2:PRINT"M E N U
  P R I N C I P A L":LOCATE10,6
:PRINT"1:- ENTRAR DADOS":LOCATE
10,8:PRINT"2:- VER DADOS":LOCAT
E10,10:PRINT"3:- GRAVAR NA FITA
"
70 LOCATE10,12:PRINT"4:- CARREG
AR DA FITA":LOCATE10,14:PRINT"5
:- OPÇÃO DE IMPRESSÃO":LOCATE10
,16:PRINT"6:- ALTERAR DADOS":LO
CATE10,18:PRINT"7:- FIM DE PROG
RAMA"
80 LOCATE10,22:PRINT"-ESCOLHA U
MA OPÇÃO-"
90 A$=INKEY$:IFA$<"1"ORA$>"7"TH
EN90
100 ONVAL(A$)GOSUB1000,2000,300
0,4000,5000,6000,7000
110 COLOR 15,4,4:GOTO60
1000 CLS:IFNU>200THENLOCATE10,1
5:PRINT"MEMÓRIA CHEIA!":BEEP
1010 GOSUB1160
1020 GOSUB1250:INPUT"DATA ";DA$
(NU)
1030 IFDA$(NU)=""THENRETURN
1040 IFLEN(DA$(NU))>8THEN1020
1050 LOCATE0,L:PRINTDA$(NU)
1060 GOSUB1250:LINEINPUT"ITEM ?
";TE$(NU):IFLEN(TE$(NU))>25THEN
1060
1070 A$=LEFT$(TE$(NU),15):LOCAT
E17-LEN(A$)/2,L:PRINTA$
1080 GOSUB1250:INPUT"QUANTIA ";
A
1090 IFA>9999999#ORA<0THEN1080
1100 LOCATE25,L:PRINTUSINGU2$;A
:AM(NU)=A
1110 GOSUB1250:INPUT"CATEGORIA
";CA$
1120 GOSUB1180:IFF=0THEN1110
1130 CA(NU)=NM:LOCATE36,L:PRINT
LEFT$(CT$(CA(NU)),3)
1140 IFNM<>8THENGT=GT+A
1150 NU=NU+1:L=L+1:IFL=19THEN10
00ELSE1020
1160 L=2:PRINT"  data";TAB(14);
"item";TAB(26);"quantia";TAB(36
)"cat";
1170 RETURN
1180 IFVAL(CA$)<>0THEN1230
1190 F=0:FORN=1TO8
1200 IFCA$=LEFT$(CT$(N),LEN(CA$
))THENF=F+1:NM=N
1210 NEXT:IFF>1THENF=0
1220 RETURN
1230 IFVAL(CA$)>8THENF=0:RETURN
1240 NM=VAL(CA$):F=1:RETURN
1250 LOCATE0,20:PRINTSPACE$(79)
;:LOCATE0,20:RETURN
2000 CLS:FORN=1TO8
2010 LOCATE10,2*N+3:PRINTN;":-
";CT$(N)
2020 NEXT
2030 TT=0:LOCATE15,23:PRINT"Qua
l categoria? ";
2040 A$=INKEY$:IFA$<"1"ORA$>"8"
THEN2040
2050 NM=VAL(A$)
2060 IFPT=1THENLPRINTTAB(40-LEN
(CT$(NM))/2);CT$(NM):LPRINTTAB(
10)"  DATA";TAB(28);"ITEM";TAB(
51);"QUANTIA"
2070 GOSUB2280
2080 FORNN=0TONU
2090 IFCA(NN)<>NMTHEN2150
2100 IF PT=1 THEN LPRINTTAB(10)
;DA$(NN);TAB(22);TE$(NN);:LPRIN
TTAB(50)USINGU2$;AM(NN)
2110 LOCATE0,L:PRINTDA$(NN);:A$
=LEFT$(TE$(NN),20):LOCATE10:PRI
NTA$;:LOCATE 30:PRINTUSINGU2$;A
M(NN);:TT=TT+AM(NN)
2120 L=L+1:IF (L=21ANDNM<>8) OR
  (L=19ANDNM=8) THEN LOCATE14,22
:PRINT"scroll? ";:ELSE GOTO 215
0
2130 A$=INKEY$:IFA$=""THEN2130
2140 GOSUB2280
2150 NEXT:IFPT=1ANDNM<>8THENLPR
INT:LPRINT:LPRINTTAB(38);USING"
TOTAL => "+U1$;TT
2160 LOCATE18,22:PRINTUSING"tot
al =>"+U1$;TT
2170 IFNM<>8THEN2250
2180 IFPT=0THEN2220
2190 LPRINT:LPRINT:LPRINTTAB(40
);USING"RENDA TOTAL =>"+U1$;TT
2200 LPRINTTAB(36);USING"DESPES
AS TOTAIS =>"+U1$;GT:LPRINTTAB(
54)"---------------"
2210 LPRINTTAB(44);USING"BALANÇ
O =>"+U1$;TT-GT
2220 LOCATE13,20:PRINTUSING"Ren
da total =>"+U1$;TT
2230 LOCATE9,21:PRINTUSING"Desp
esas totais =>"+U1$;GT
2240 LOCATE17,22:PRINTUSING"Bal
anço =>"+U1$;TT-GT;
2250 A$=INKEY$:IFA$=""THEN2250
2260 IFA$<>CHR$(13)THEN2000
2270 RETURN
2280 L=3:CLS:COLOR 15,NM+2:LOCA
TE20-LEN(CT$(NM))/2:PRINTCT$(NM
)
2290 PRINT"  data";TAB(18)"item
";TAB(32)"quantia";
2300 FORN=3TO20
2310 VPOKEN*40+10,22:VPOKEN*40+
```



```
30,22
2320 NEXT:RETURN
3000 CLS:MOTOR:LOCATE0,10:PRINT
"Posicione a fita e pressione <
RETURN>"
3010 IFINKEY$=""THEN3010
3020 MOTOR:PRINT"Tecle <REC/PLA
Y> e pressione <RETURN>"
3030 IFINKEY$=""THEN3030
3040 LOCATE8,15:INPUT"Nome do a
rquivo ";DA$:AR$="CAS:"+DA$
3050 OPEN AR$ FOR OUTPUT AS#1
3060 PRINT#1,NU
3070 FORN=0TONU-1
3080 PRINT#1,DA$(N);",";TE$(N);
",";AM(N),CA(N)
3090 NEXT:CLOSE#1:RETURN
4000 CLS:LOCATE8,10:INPUT"Nome
do arquivo ";DA$:AR$="CAS:"+DA$
4010 MOTOR:LOCATE0,12:PRINT"Pos
icione a fita e tecle <RETURN>"
4020 IFINKEY$=""THEN4020
4030 MOTOR:LOCATE0,14:PRINT"Pre
ssione a tecla <PLAY> e <RETURN
>"
4040 IFINKEY$=""THEN4040
4050 MOTOR:GT=0:OPEN AR$ FOR IN
PUT AS#1
4060 LOCATE8,18:PRINT"Achei ";D
A$
4070 INPUT#1,NU
4080 FORN=0TONU-1
4090 INPUT#1,DA$(N),TE$(N),AM(N
),CA(N)
4100 IFCA(N)<>8THENGT=GT+AM(N)
4110 NEXT:CLOSE#1:RETURN
5000 CLS:LOCATE10,10:PRINT"Impr
essora: "
5010 LOCATE15,12:PRINT"[L]igada
"
5020 LOCATE15,13:PRINT"[D]eslig
ada"
5030 A$=INKEY$:IFA$=""THEN5030
5040 IFA$="L"THENPT=1:RETURN
5050 IFA$="D"THENPT=0ELSE5030
5060 RETURN
6000 IFNU=0THENRETURN
6010 CLS:PRINT"  data";TAB(14);
"item";TAB(26);"quantia";TAB(36
);"cat"
6020 LOCATE0,20:PRINT"Pressione
 [=>] para avançar"
6030 LOCATE0,21:PRINT"Pressione
 [<=] para retroceder":PRINT"Ou
 a barra de espaços para editar
":GOTO6080
6040 A$=INKEY$:IFA$=""THEN6040
6050 IFASC(A$)=29ANDM>0THENM=M-
1:GOTO6080
6060 IFASC(A$)=28ANDM<NU-1THENM
=M+1:GOTO6080
6070 IFASC(A$)=32THEN6100ELSE60
40
6080 LOCATE0,3:PRINTDA$(M);TAB(
10);LEFT$(TE$(M),15);:PRINTTAB(
26);USINGU2$;AM(M);:PRINTTAB(36
);LEFT$(CT$(CA(M)),3)
6090 GOTO 6040
6100 IFCA(M)<>8THENGT=GT-AM(M)
6110 D$="":PRINT:INPUT"Data ";D
$:IFD$=""THEN6130
6120 IFLEN(D$)>8THEN6110ELSEDA$
(M)=D$
6130 D$="":PRINT:INPUT"Item ";D
$:IFD$=""THEN6150
6140 IFLEN(D$)>25THEN6130ELSETE
$(M)=D$
6150 PRINT:INPUT"Quantia ";A:IF
A=0THEN6170
6160 IFA<00RA>9999999#THEN6150E
LSEAM(M)=A
6170 CA$="":PRINT:INPUT"Categor
ia ";CA$:IFCA$=""THEN6200
6180 GOSUB1180:IFF=0THEN6170
6190 CA(M)=NM
6200 IFCA(M)<>8THENGT=GT+AM(M)
6210 RETURN
7000 CLS:LOCATE10,15:PRINT"Fim
do programa?"
7010 A$=INKEY$:IFA$=""THEN 7010
7020 IFA$<>"S"THENRETURN
```

```
10  REM  ONERR  GOTO 8000
20  DIM TE$(200),QT(200),DT$(20
0),CT$(8),CT(200)
25  FOR N = 1 TO 7: READ K$: NE
XT
30  FOR N = 1 TO 8: READ CT$(N)
: NEXT
35  DATA  ENTRAR DADOS,VER DADO
S,GRAVAR EM DISCO,CARREGAR DO D
ISCO,IMPRIMIR,ALTERAR UM DADO,F
IM DE PROGRAMA
40  DATA  MANUTENCAO DA CASA,L
AZER,ALUGUEIS E TAXAS,ROUPAS,AU
TOMOVEL,FERIAS,MISCELANEA,RENDA
S
50  HOME : RESTORE : INVERSE :
VTAB 2: HTAB 7: PRINT " M E N U
   P R I N C I P A L "
60  FOR N = 1 TO 7: READ K$
70  HTAB 10: VTAB N * 2 + 4: PR
INT N;":- ";K$
80  NEXT
90  VTAB 23: HTAB 20: PRINT " O
PCAO => ";: GET A$
100  IF A$ < "1" OR A$ > "7" TH
EN 90
110  PRINT A$;: NORMAL : HOME
120  ON  VAL (A$) GOSUB 1000,20
00,3000,4000,5000,6000,7000
130  GOTO 50
1000  IF NU > 200 THEN  VTAB 15
: HTAB 10: PRINT  CHR$ (7);"MEM
ORIA CHEIA!"; CHR$ (7): FOR N =
 1 TO 1000: NEXT : RETURN
1010  GOSUB 1160
1020  GOSUB 1250: INPUT "DATA (
DD/MM/AA): ";DT$(NU)
1030  IF DT$(NU) = "" THEN  RET
URN
1040  IF  LEN (DT$(NU)) > 8 THE
N 1020
1050  VTAB L: PRINT DT$(NU)
1060  GOSUB 1250: INPUT "ITEM:
";TE$(NU): IF  LEN (TE$(NU)) >
25 THEN 1060
1070 A$ =  LEFT$ (TE$(NU),15):
VTAB L: HTAB 17 -  LEN (A$) / 2
: PRINT A$
1080  GOSUB 1250: INPUT "QUANTI
A: ";QT(NU)
1090  IF QT(NU) > 99999999 OR Q
T(NU) < 0 THEN 1080
1100  VTAB L: HTAB 26: PRINT QT
(NU)
1110  GOSUB 1250: INPUT "CATEGO
RIA: ";C$
1120 D$ = C$: GOSUB 9000: IF F
= 0 THEN 1110
1130 CT(NU) = NM: VTAB L: HTAB
```

```
38: PRINT  LEFT$ (CT$(CT(NU)),3
);
1140  IF C <  > 8 THEN GT = GT
+ QT(NU)
1150 NU = NU + 1:L = L + 1: IF
L = 21 THEN 1000
1155  GOTO 1020
1160 L = 3: PRINT "  data"; TAB
( 15);"item"; TAB( 26);"quantia
"; TAB( 38);"cat";
1170  RETURN
1250  HTAB 1: VTAB 23: CALL  -
958: RETURN
2000  HOME : FOR N = 1 TO 8
2010  HTAB 10: VTAB 2 * N + 3
2020  PRINT N;":- ";CT$(N): NEX
T
2030 TT = 0: HTAB 15: VTAB 23:
PRINT "Qual categoria? ";
2040  GET A$: IF A$ < "1" OR A$
 > "8" THEN 2040
2050 NM =  VAL (A$)
2060  IF PT = 1 THEN  PRINT : P
RINT  CHR$ (4);"PR#1": PRINT  C
HR$ (9);"80N": PRINT  SPC( 40 -
  LEN (CT$(NM)) / 2);CT$(NM): P
RINT : PRINT  SPC( 10);"  DATA"
; SPC( 17);"ITEM"; SPC( 18)"QUA
NTIA": PRINT  CHR$ (4);"PR#0"
2070  GOSUB 2280
2080  FOR NN = 0 TO NU
2090  IF CT(NN) <  > NM THEN 21
50
2100  IF PT = 1 THEN  PRINT : P
RINT  CHR$ (4);"PR#1": PRINT  C
HR$ (9);"80N";: PRINT  SPC( 10)
;DT$(NN); SPC( 5 + (8 -  LEN (D
T$(NN))));TE$(NN); SPC( 6 + 25
-  LEN (TE$(NN))); SPC( 7 -  LE
N ( STR$ (QT(NN))));QT(NN): PRI
NT  CHR$ (4);"PR#0"
2110  VTAB L: HTAB 1: PRINT DT$
(NN);:A$ =  LEFT$ (TE$(NN),20):
 HTAB 10: PRINT A$;: HTAB 32 +
(7 -  LEN ( STR$ (QT(NN)))): PR
INT QT(NN);:TT = TT + QT(NN)
2120 L = L + 1: IF (L = 21 AND
NM <  > 8) OR (L = 19 AND NM =
8) THEN  VTAB 23: HTAB 15: PRIN
T "Scroll ?";: GET A$: GOSUB 22
80
2150  NEXT : IF PT = 1 AND NM <
  > 8 THEN  PRINT : PRINT  CHR$
 (4);"PR#1": PRINT  CHR$ (9);"8
0N": PRINT : PRINT : PRINT  SPC
( 40);"TOTAL =>";TT: PRINT  CHR
$ (4);"PR#0"
2160  VTAB 23: HTAB 21: PRINT "
TOTAL =>";TT;
2170  IF NM <  > 8 THEN 2250
2180  IF PT = 0 THEN 2220
2190  PRINT : PRINT  CHR$ (4);"
PR#1": PRINT  CHR$ (9);"80N": P
RINT : PRINT : PRINT  SPC( 40);
"RENDA TOTAL =>"; SPC( 9 -  LEN
 ( STR$ (TT)));TT: PRINT  SPC(
36);"DESPESAS TOTAIS =>"; SPC(
9 -  LEN ( STR$ (GT)));GT
2200  PRINT  SPC( 54);"--------
-"
2210  PRINT  SPC( 44);"BALANCO
=>"; SPC( 9 -  LEN ( STR$ (TT -
GT)));TT - GT: PRINT  CHR$ (4)
;"PR#0"
2220  VTAB 21: HTAB 18: PRINT "
RENDA TOTAL =>";TT
2230  VTAB 22: HTAB 15: PRINT "
DESPESAS TOTAIS =>";GT;
2240  VTAB 23: HTAB 20: PRINT "
BALANCO =>";TT - GT;
2250  GET A$
2260  IF A$ <  >  CHR$ (13) THE
N 2000
2270  RETURN
2280  HOME :L = 4: HTAB 20 -  L
EN (CT$(NM)) / 2: PRINT CT$(NM)
2290  PRINT "  data"; TAB( 18);
"item"; TAB( 32);"quantia";
2300  FOR N = 4 TO 21: VTAB N:
HTAB 9: PRINT "!";: HTAB 31: PR
INT "!";: NEXT
2310  RETURN
3000  HOME : VTAB 15: HTAB 10:
INPUT "Nome do arquivo? ";DA$
3010  IF DA$ = "" THEN  RETURN
3020  PRINT : PRINT  CHR$ (4);"
OPEN";DA$
3030  PRINT  CHR$ (4);"DELETE";
DA$
3040  PRINT  CHR$ (4);"OPEN";DA
$
3050  PRINT  CHR$ (4);"WRITE";D
A$
3060  PRINT NU
3070  FOR N = 0 TO NU - 1: PRIN
T DT$(N); CHR$ (13);TE$(N); CHR
$ (13);QT(N); CHR$ (13);CT(N):
NEXT
3080  PRINT  CHR$ (4);"CLOSE":
RETURN
4000  HOME : VTAB 15: HTAB 10:
INPUT "Nome do arquivo? ";DA$
4010  IF DA$ = "" THEN  RETURN
4015 GT = 0
4020  PRINT : PRINT  CHR$ (4);"
OPEN";DA$
4030  PRINT  CHR$ (4);"READ";DA
$
4040  INPUT NU
4050  FOR N = 0 TO NU - 1
4060  INPUT DT$(N),TE$(N),QT(N)
,CT(N)
4065  IF CT(N) <  > 8 THEN GT =
 GT + QT(N)
4070  NEXT
4080  PRINT  CHR$ (4);"CLOSE"
4090  RETURN
5000  HOME : VTAB 15: HTAB 10:
PRINT "Ligo a impressora? (S/N)
";
5010  GET A$: IF A$ <  > "S" AN
D A$ <  > "N" THEN 5010
5020  PRINT : PRINT  TAB( 20);"
OK!": IF A$ = "N" THEN PT = 0:
RETURN
5030 PT = 1: RETURN
6000  IF NU = 0 THEN  RETURN
6020  VTAB 20: PRINT : PRINT "P
ressione [->] para avancar": PR
INT "Pressione [<-] para retroc
eder": PRINT "Ou a barra de esp
acos para corrigir";: GOTO 6080
6030  GET A$
6040  IF A$ =  CHR$ (32) THEN 6
100
6050  IF A$ =  CHR$ (21) AND M
< NU - 1 THEN M = M + 1: GOTO 6
080
6060  IF A$ =  CHR$ (8) AND M >
 0 THEN M = M - 1: GOTO 6080
6070  GOTO 6030
6080  VTAB 5: HTAB 1: CALL  - 8
68: PRINT "DATA:       ";DT$(M)
6090  VTAB 8: CALL  - 868: PRIN
T "ITEM:       ";TE$(M): VTAB 11
: CALL  - 868: PRINT "QUANTIA:
  ";QT(M): VTAB 14: CALL  - 868
: PRINT "CATEGORIA: ";CT$(CT(M)
)
6095  GOTO 6030
6100  IF CT(M) <  > 8 THEN GT =
 GT - QT(M)
6110  FOR X = 1 TO 4
6120  VTAB X * 3 + 2: HTAB 12:
CALL  - 868: INPUT "";D$
6130  ON X GOTO 6200,6300,6400,
6500
6140  NEXT
6150  IF CT(M) <  > 8 THEN GT =
 GT + QT(M)
6160  RETURN
6200  IF D$ = "" THEN D$ = DT$(
M): GOTO 6600
6210  IF  LEN (D$) > 8 THEN 612
0
6220 DT$(M) = D$: NEXT
6300  IF D$ = "" THEN D$ = TE$(
M): GOTO 6600
6310  IF  LEN (D$) > 25 THEN 61
20
6320 TE$(M) = D$: NEXT
6400  IF  VAL (D$) = 0 THEN D$
=  STR$ (QT(M)): GOTO 6600
6410  IF  VAL (D$) > 9999999 TH
EN 6120
6420 QT(M) =  VAL (D$): NEXT
6500  IF D$ = "" THEN D$ = CT$(
CT(M)): GOTO 6600
6510  GOSUB 9000: IF F = 0 THEN
 6120
6520 CT(M) = NM: GOTO 6140
6600  VTAB 3 * X + 2: HTAB 12:
PRINT D$: NEXT : RETURN
7000  VTAB 15: HTAB 8: PRINT "T
ermino o programa? (S/N)";
7010  GET A$: IF A$ <  > "S" AN
D A$ <  > "N" THEN 7010
7020  IF A$ = "N" THEN  RETURN
7030  END
8000  IF  PEEK (222) <  > 5 THE
N  RESUME
8005  PRINT  CHR$ (4);"CLOSE"
8010  PRINT  CHR$ (4);"DELETE";
DA$
8020  HOME : VTAB 20: HTAB 10:
FLASH : PRINT "ARQUIVO INEXISTE
NTE!"
8030  FOR I = 1 TO 2000: NEXT :
 NORMAL
8040  GOTO 50
9000  IF  VAL (D$) <  > 0 THEN
9030
9010 F = 0: FOR N = 1 TO 8
9020  IF  LEFT$ (D$,3) =  LEFT$
 (CT$(N),3) THEN F = 1:NM = N
9025  NEXT : RETURN
9030  IF  VAL (D$) > 8 THEN F =
 0: RETURN
9040 NM =  VAL (D$):F = 1: RETU
RN
```

CONCURSO INPUT

2º Sorteio

5 microcomputadores para os sorteados do 1º ao 5º prêmios da Loteria Federal

REGULAMENTO

Para concorrer no 2º sorteio do sensacional Concurso INPUT, você deve juntar 06 selos que virão no canto inferior direito dos fascículos 1 a 16 (1 selo por edição), indistintamente. Se você enviou a cartela para o 1º sorteio, ainda sobraram 02 selos que poderão ser usados agora. Caso prefira não usá-los, envie os selos das demais edições citadas.

Cole os 06 selos na cartela abaixo, ou envie-os colados num papel dentro de um envelope.

Você receberá um número com o qual concorrerá no sorteio da Loteria Federal do dia 04 de outubro de 1986. Serão aceitas as cartelas/selos enviados até 05.09.86.

A cartela/selos deverão ser enviados juntamente com seu nome, endereço, telefone, nº da carteira de identidade (do participante ou responsável) e CPF, num envelope endereçado ao CONCURSO INPUT - Caixa Postal 9442, CEP 01000, São Paulo - SP.

OBS.: A prescrição do direito aos prêmios se dará 180 dias após o sorteio. O resultado será publicado nos fascículos INPUT e os prêmios entregues por nossos distribuidores nas cidades de residência dos contemplados.

Se tiver alguma dúvida, consulte nosso Serviço de Atendimento ao Leitor, pelo telefone (011) 815-8055 - ramal 235.

INPUT

Nome:

Endereço:

CEP: Cidade:

Estado: Fone: Cédula de Identidade:

CPF:

Cert. Aut. Min. Faz. nº 01/00/237/86

Não perca mais esta chance de ganhar um dos 10 sensacionais microcomputadores HOT BIT HB 8.000 SHARP.
Continue colecionando INPUT. Seu micro ganha vida.

NOVA CULTURAL

NO PRÓXIMO NÚMERO

PROGRAMAÇÃO BASIC
Facilite a compreensão das instruções, listas e tabelas, organizando corretamente a disposição das informações.

PROGRAMAÇÃO DE JOGOS
Como tornar mais difícil um jogo de labirinto. Marque o tempo necessário para encontrar um tesouro.

CÓDIGO DE MÁQUINA
Os problemas com binários e hexadecimais quando é necessário representar números negativos.

PROGRAMAÇÃO BASIC
Técnicas para tornar os programas mais curtos. Os comandos abreviados.